DOCUMENT RESUME

ED 198 375 CE 028 094

TITLE	Military Curricula for Vocational & Technical Education. Programmer/Analyst 4-4.
INSTITUTION	Department of the Army, Washington, D.C.; Ohio State Univ., Columbus. National Center for Research in Vocational Education.
SPONS AGENCY	Bureau of Occupational and Adult Education (DHEW/OE), Washington, D.C.
PUB DATE	Sep 78
NOTE	711p.; Not available in paper copy due to small and broken print.
EDRS PRICE	MF04 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	Behavioral Objectives; *Computer Programs; Computers; *Computer Science Education; Course Descriptions; Curriculum Guides; Learning Activities; Postsecondary Education; Programed Instructional Materials; Programers; *Programing; *Programing Languages; Secondary Education; Technical Education
IDENTIFIERS	*COBOL Programming Language; Military Curriculum Project

ABSTRACT

This program of instruction and various instructional materials for a secondary-postsecondary level course for programmer/analysts is one of a number of military-developed curriculum packages selected for adaptation to vocational instruction and curriculum development in a civilian setting. The eight-week, three-section course is designed to provide the skill to program electronic computers structured in COBOL and to code job streams using IBM, DOS, JCL, and utility programs. The program of instruction suggests a time schedule and gives the learning objective and reference(s) for each topic. Section 1, Data Representation, consists of a programmed text covering the binary, octal, and lexadecimal systems. Section 2, Basic COBOL Programming, contains a text and problem exercises with some answers. Topic areas include processing and updating a sequential file, producing an edited report and a report with calculations, processing external and internal tables, and debugging syntax errors. Section 3, Operating Systems, includes a text and programmed text. It focuses on disk operating systems (DOS) organization and operation. Specific topics include coding DOS Job Streams, DOS Librarian Programs, and DOS Utilities and Sorts. (YLB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED198375

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY.

This military technical training course has been selected and adapted by The Center for Vocational Education for "Trial Implementation of a Model System to Provide Military Curriculum Materials for Use in Vocational and Technical Education," a project sponsored by the Bureau of Occupational and Adult Education, U.S. Department of Health, Education, and Welfare.

MILITARY CURRICULUM MATERIALS

The military-developed curriculum materials in this course package were selected by the National Center for Research in Vocational Education Military Curriculum Project for dissemination to the six regional Curriculum Coordination Centers and other instructional materials agencies. The purpose of disseminating these courses was to make curriculum materials developed by the military more accessible to vocational educators in the civilian setting.

The course materials were acquired, evaluated by project staff and practitioners in the field, and prepared for dissemination. Materials which were specific to the military were deleted, copyrighted materials were either omitted or approval for their use was obtained. These course packages contain curriculum resource materials which can be adapted to support vocational instruction and curriculum development.

The National Center Mission Statement

The National Center for Research in Vocational Education's mission is to increase the ability of diverse agencies, institutions, and organizations to solve educational problems relating to individual career planning, preparation, and progression. The National Center fulfills its mission by:

- Generating knowledge through research
- Developing educational programs and products
- Evaluating individual program needs and outcomes
- Installing educational programs and products
- Operating information systems and services
- Conducting leadership development and training programs

FOR FURTHER INFORMATION ABOUT
Military Curriculum Materials
WRITE OR CALL
Program Information Office
The National Center for Research in Vocational Education
The Ohio State University
1960 Kenny Road, Columbus, Ohio 43210
Telephone: 614/486-3655 or Toll Free 800/848-4815 within the continental U.S. (except Ohio)

THE NATIONAL CENTER
FOR RESEARCH IN VOCATIONAL EDUCATION
The Ohio State University · 1960 Kenny Road · Columbus, Ohio 43210
Tel: (614) 486-3655 Cable: CTVOCEDOSU/Columbus, Ohio

Military Curriculum Materials for Vocational and Technical Education

Information and Field Services Division

The National Center for Research in Vocational Education

Military Curriculum Materials Dissemination Is . . .

an activity to increase the accessibility of military-developed curriculum materials to vocational and technical educators.

This project, funded by the U.S. Office of Education, includes the identification and acquisition of curriculum materials in print form from the Coast Guard, Air Force, Army, Marine Corps and Navy.

Access to military curriculum materials is provided through a "Joint Memorandum of Understanding" between the U.S. Office of Education and the Department of Defense.

The acquired materials are reviewed by staff and subject matter specialists, and courses deemed applicable to vocational and technical education are selected for dissemination.

The National Center for Research in Vocational Education is the U.S. Office of Education's designated representative to acquire the materials and conduct the project activities.

Project Staff:

Wesley E. Budke, Ph.D., Director
National Center Clearinghouse

Shirley A. Chase, Ph.D.
Project Director

What Materials Are Available?

One hundred twenty courses on microfiche (thirteen in paper form) and descriptions of each have been provided to the vocational Curriculum Coordination Centers and other instructional materials agencies for dissemination.

Course materials include programmed instruction, curriculum outlines, instructor guides, student workbooks and technical manuals.

The 120 courses represent the following sixteen vocational subject areas:

- Agriculture
- Aviation
- Building & Construction Trades
- Clerical Occupations
- Communications
- Drafting
- Electronics
- Engine Mechanics
- Food Service
- Health
- Heating & Air Conditioning
- Machine Shop
- Management & Supervision
- Meteorology & Navigation
- Photography
- Public Service

The number of courses and the subject areas represented will expand as additional materials with application to vocational and technical education are identified and selected for dissemination.

How Can These Materials Be Obtained?

Contact the Curriculum Coordination Center in your region for information on obtaining materials (e.g., availability and cost). They will respond to your request directly or refer you to an instructional materials agency closer to you.

CURRICULUM COORDINATION CENTERS

EAST CENTRAL
Rebecca S. Douglass
Director
100 North First Street
Springfield, IL 62777
217/782-0759

NORTHWEST
William Daniels
Director
Building 17
Airdustrial Park
Olympia, WA 98504
206/753-0879

MIDWEST
Robert Patton
Director
1515 West Sixth Ave.
Stillwater, OK 74704
405/377-2000

SOUTHEAST
James F. Shill, Ph.D.
Director
Mississippi State University
Drawer DX
Mississippi State, MS 39762
601/325-2510

NORTHEAST
Joseph F. Kelly, Ph.D.
Director
225 West State Street
Trenton, NJ 08625
609/292-6562

WESTERN
Lawrence F. H. Zane, Ph.D.
Director
1776 University Ave.
Honolulu, HI 96822
808/948-7834

PROGRAMMER/ANALYST COURSE

Classroom Course 4-4

532-74F1

Developed by:

United States Army

Development and Review Dates:

September 1978

Occupational Area:

Clerical Occupations

Target Audiences:

Grade 12 - Adult

Print Pages: 625

Microfiche: 11

Availability:

Vocational Curriculum Coordination Centers

Contents:	Type of Materials: Lesson Plans:	Programmed Text:	Student Workbook:	Handouts:	Text Materials:	Audio-Visuals:	Instructional Design: Performance Objectives:	Tests:	Review Exercises:	Additional Materials Required:	Type of Instruction: Group Instruction:	Individualized:
Data Representation	●	●					●	●			●	●
Basic Cobol Programming	●			●	●		●	●	●		●	
Operating Systems	●			●	●		●		●		●	

X Materials are recommended but not provided.

Course Description:

This course is designed to provide the skill to program electronic computers structured in Cobol and to code jobstreams using IBM, DOS, JCL, and utility programs. The course covers 8 weeks of instruction.

Data Representation consists of a programmed text covering the binary, octal and lexadecimal number systems.

Basic Cobol Programming section contains a text and problem exercises with some having answers. Topic areas include processing and updating a sequential file; producing an edited report and a report with calculations; processing external and internal tables; debugging syntax errors.

The Operating Systems segment focuses on disk operating systems organization and operation. Specific topics of coding DOS Jobstreams, DOS Librarian Programs, DOS Utilities and Sorts are included.

Programmer/Analyst Course
532-74F1

Classroom Course 4-4

TABLE OF CONTENTS

	Page
Program of Instruction	1
Data Representation Programmed Text	20
Basic Cobol Programming	148
Operating Systems	473

Annex F - Update a Sequential Master File Using a Sequential Transaction File

Terminal Learning Objective: Given program specifications, write a COBOL program to update a sequential master file with a transaction file containing adds, deletes, changes and errors. The program must also produce an update listing and a master file listing.

532-74F-1242	Establish Read Switches	Type
Hours -	2	2C

Learning Objective: Using working storage, establish read switches to govern the access to each file.

Ref: Shelley/Cashman Structured COBOL

532-74F-1243	Update Master File	Type
Hours -	29	7C, 22PE2

Learning Objective: Given program specifications, match records on two files to produce an updated output master.

Ref: Shelley/Cashman Structured COBOL

532-74F-1244	COBOL V Review and Exam	Type
Hours -	8	4C, 4E

Learning Objective: N/A

Ref: N/A

Annex F Total Hours Allotted: 39

Annex E - Process External and Internal Tables in COBOL

Terminal Learning Objective: Given program specifications code COBOL routines to build tables from external or internal data and match input files to the completed table.

532-74F-1238	Establish Internal Table	Type
Hours -	14	2C, 12PE2

Learning Objective: Using a month conversion table, write the COBOL statements to change the numeric month to the month's name.

Ref: Shelley/Cashman Structured COBOL

532-74F-1239	External Table Handling	Type
Hours -	14	5.5C, 8.5PE2

Learning Objective: Given program specifications, code COBOL routines to build a table and search the table for match or no match condition.

Ref: Shelley/Cashman Structured COBOL

532-74F-1240	COBOL IV Exam and Review/Critique	Type
Hours -	6	2C, 4E

Learning Objectives: N/A

Ref: N/A

Annex E Total Hours Allotted; 34

532-74F-1234	COBOL III Review and Exam	Type
Hours -	8	4C, 4E

Learning Objective: N/A

Ref: N/A

Annex D Total Hours Allotted: 37

Annex D - Produce a Report with Calculations and Minor Control Breaks in COBOL

Terminal Learning Objective: Given program specifications, code a COBOL program that will perform arithmetic calculations and test for changes in input data to produce a report with group totals at a control break.

532-74F-1230	Test for Control Breaks	Type
Hours -	12.5	2.5C, 10PE2

Learning Objective: Given input control fields, code routines to process changes in the fields that will result in recognizing the change and processing each group of like items as individual entities.

Ref: Shelley/Cashman Structured COBOL

532-74F-1232	Advanced Condition Tests	Type
Hours -	2	2C

Learning Objective: Given specifications, code routines that use condition names and nested IF statements syntactically and logically correct.

Ref: Shelley/Cashman Structured COBOL

532-74F-1232	Arithmetic Verbs	Type
Hours -	6	1.5. 4.5PE2

Learning Objective: Given specifications, code COBOL arithmetic verbs in conjunction with condition statements to produce minor totals at control breaks.

Ref: Shelley/Cashman Structured COBOL

532-74F-1233	Page headers & Trailers	Type
Hours -	8.5	2.5C, 6PE2

Learning Objective: Given report requirements, code COBOL routines that will result in multiple headings with the current data and sequential page numbers on output reports.

Ref: Shelley/Cashman Structured COBOL

532-74F-1224	COBOL II Review & Exam	Type
Hours -	6	2C, 4E

Learning Objective: N/A

Ref: N/A

Annex C Total Hours Allotted: 39

6

Annex C - Produce Edited Report with Minor Control Breaks in COBOL

Terminal Learning Objective: Given program specifications, code a COBOL program to read sequential input, perform condition tests, accumulate totals and use report editing to produce a report with a heading and total line.

532-74F-1220 Code Special Names Paragraph Type

Hours - .5 .5C

Learning Objective: Given the COBOL manual, code the special names paragraph correctly.

Ref: N/A

532-74F-1221 Establishing a Heading Line and Accumulations in the Data Division Type

Hours - 1.5 1.5C

Learning Objective: Given program specifications code the Data Division entries to establish a heading line and accumulators.

Ref: Shelley/Cashman Structured COBOL

532-74F-1222 Code ADD and IF Verbs Logically Within the Procedure Division Type

Hours - 14 4C, 10PE2

Learning Objective: Given program specifications, code the COBOL routines to print a heading, accumulate totals and to conditional tests.

Ref: Shelley/Cashman Structured COBOL

532-74F-1223 Editing Data and Formatting A Total Line Type

Hours - 17 5C, 12PE2

Learning Objective: Given program specifications, write COBOL routines to edit numeric data and print a final total line.

Ref: Shelley/Cashman Structured COBOL

532-74F-1214	Code Simple COBOL Program	Type
Hours -	10	1C, 9PE2

Learning Objective: Given program specifications, code a program to read a sequential file, move data and print a simple report.

Ref: Shelly/Cashman Structured COBOL

532-74F-1215	COBOL I Exam & Review/Critique	Type
Hours -	7	3C, 4E

Learning Objective: N/A

Ref: N/A

Annex B Total Hours Allotted: 37

4

Annex B - Process Sequential File in COBOL

532-74F-1210	Code COBOL Identification Division	Type
Hours -	1.5	1C, .5PE2

Learning Objective: Given program specifications, code a COBOL Identification Division that is syntactically correct and that meets specifications.

Ref: Shelley/Cashman Structured COBOL

532-74F-1211	Code COBOL Environment Division	Type
Hours -	1.5	1C, .5PE2

Learning Objective: Given program specifications, code a COBOL Environment Division that is syntactically correct and that meets specifications.

Ref: Shelley/Cashman Structured COBOL

532-74F-1212	Code COBOL Data Division	Type
Hours -	8	4C, 4PE2

Learning Objective: Given program specifications, code a COBOL Data Division that is syntactically correct and that contains all entries necessary for correct program operation.

Ref: Shelly/Cashman Structured COBOL

532-74F-1213	Code I/O and Movements Verbs	Type
Hours -	9	9C

Learning Objective: Given program specifications, code the COBOL verbs necessary to perform sequential I/O operations and to move data within memory.

Ref: Shelly/Cashman Structured COBOL

532-74F-1205	Problem Solving Exam & Review	Type
Hours -	7	3C, 4E

Learning Objective: N/A

Ref: N/A

Annex A Total Hours Allotted: 41

3

SECTION IV - ANNEXES

Annex A - Solve ADP Problems

Terminal Learning Objectives: Given a narrative problem, construct a systems flowchart, structured flowchart, HIPO and IPO that will represent a solution to the problem in structured terms.

531-74F-1201	Draw a Systems Flowchart	Type
Hours -	1.5	.5C, 1PE2

Learning Objective: Given the inputs and outputs to a program, draw a chart that will use the proper symbols to depict the flow of data in the system.

Ref: Class Notes

531-74F-1202	Draw a Hierarchy Chart	Type
Hours -	5	3C, 2PE2

Learning Objective: Given program specifications, draw a hierarchy chart that correctly reflects the logical structure of a program that will result in the required output.

Ref: Class Notes, Check Problem

532-74F-1203	Write an IPO Chart	Type
Hours -	15.5	6C, 9.5PE2

Learning Objective: Given a HIPO and program specifications correctly determine the inputs, outputs and processing required to meet the requirements.

Ref: Class Notes, PE 1 through PE 4

532-74F-1204	Draw a Structured Flowchart	Type
Hours -	12	4C, 8PE2

Learning Objective: Given program specifications, a HIPO and an IPO draw a structured flowchart that represents the logic flow within an IPO.

Ref: Class Notes, PE1 through PE4

2

A. Course: 532-74F1, Programmer/Analyst

B. Purpose: To provide enlisted personnel with the ability to program electronic computers in structured COBOL; to code job streams using IBM DOS JCL and utility programs and to work in the Army ADP environment interfacing with USACSC and other ADP agencies. Personnel are trained in MOS 74F10

C. Prerequisites:

1. Must have a course in high school algebra or score 45 or greater on GED test 5, high school level.

2. Standard score of 105 or better in aptitude area ST or CL.

3. Nine months or more of active duty remaining after course completion.

4. No security clearance required for this course.

5. AR 611-201, Special Requirements Section, requires a security clearance for MOS 74F.

D. Scope: Students learn ADP topics through in-class conference, practical exercises in COBOL, JCL and utilities and self-paced instruction.

E. Length:	Peacetime	Mobilization
	9 weeks, 3 days	8 weeks, 3 days

F. Training Location: US Army Institute of Administration
Fort Benjamin Harrison, Indiana 46216

G. MOS Feeder pattern:

Prerequisite MOS	MOS trained in this course	Feeds Following MOS
09B	74F10	74F20, 74F30, 74F40, 74Z50

H. Ammunition Requirements: None.

UNITED STATES ARMY INSTITUTE OF ADMINISTRATION
Fort Benjamin Harrison, Indiana 46216

PROGRAM OF INSTRUCTION
FOR
PROGRAMMER/ANALYST COURSE
532-74F1

Length: Peacetime - 9 weeks, 3 days
Mobilization - 8 weeks, 3 days

Approved by:

121-012-1400-190-A

11

Annex G - Data Representation

Terminal Learning Objective: Given the decimal binary and hexadecimal numbering systems, be able to describe the properties and manipulations of each and be able to manipulate the numbers within each number system.

531-74F-1245	Manipulate Number Systems	Type
Hours -	2	1C, 1PE2

Learning Objective: Given several problems in the decimal, binary, and hexadecimal number systems, perform the indicated manipulations for each.

Ref: Data Representation Handout

531-74F-1246	Exam and Review/Critique	Type
Hours -	2	2E

Learning Objective: N/A

Ref: N/A

Annex G Total Hours Allotted: 4

Annex H - Debug COBOL Syntax Errors

Terminal Learning Objective: Given a COBOL listing with three different levels of snytax errors, determine all statements in error and correct them.

531-74F-1247	Determine Erroneous Statement	Type
Hours -	3	1.5C, 1.5PE2

Learning Objective: Given a COBOL diagnostic and source listing, find all language syntax errors and correct them.

Ref: Shelley/Cashman Structured COBOL

531-74F-1248	Exam and Review/Critique	Type
Hours -	1	1E

Learning Objective: N/A

Ref: N/A

Annex H Total Hours Allotted: 4

Annex I - Debug DOS Program Abends

Terminal Learning Objective: Given a COBOL compile listing with all options, a linkedit map and a core dump, find the error causing the program to abend and correct it.

531-74F-1250	Code a Simple ALC Program	Type
Hours -	8	6C, 2PE2

Learning Objective: Given in-class notes, code and execute an ALC program to produce a calendar.

Ref: ALC handout

531-74F-1251	Correct DOS Abend	Type
Hours -	10	3C, 7PE2

Learning Objective: Given compile and linkedit listings, and a core dump, find a statement in error and correct it.

Ref: PE handouts

531-74F-1252	Exam and Review/Critique	Type
Hours -	4	2C, 2E

Learning Objectives: N/A

Ref: N/A

Annex I Total Hours Allotted: 22

Annex J - Know DOS Organization and Operation

Terminal Learning Objective: Given the System Control and Service Manual, describe the function of each component of DOS as well as the benefits of operating systems and the effects of multiprogramming.

532-74F-1255	Operating Systems	Type
Hours -	2	2C

Learning Objective: Given class reference materials, describe the purpose and advantages of an operating system naming at least 4 advantages.

Ref: OS Handout, System Control and Service

532-74F-1256	Multiprogramming Concepts	Type
Hours -	2	2C

Learning Objective: With class notes, describe the effect multiprogramming has on total system throughput.

Ref: OS Handout

532-74F-1257	DOS Libraries	Type
Hours -	2	2C

Learning Objective: Given class references and a list of programs, state which DOS library will store each program.

Ref: OS Handout
System Control and Service
DOS Job Control, Shelly/Cashman

Annex J Total Hours Allotted: 6

Annex K - Code DOS Jobstream

Terminal Learning Objective: Given all class reference material, program specifications and job requirements, code jobstreams to compile link and/or execute programs in the DOS or DOS-E environment.

532-74F-1260	Code DOS Jobstream	Type
Hours -	16.5	9.5C, 7PE2

Learning Objective: Given all class reference material, program specifications and job requirements, code JCL to compile, link and/or execute programs using unit record and utility I/O devices. The JCL must be free of syntax errors and must meet all requirements.

Ref: OS Handout
System Control & Service
DOS Job Control, Shelley/Cashman

532-74F-1261	Code DOS-E Extensions to DOS	Type
Hours -	3.5	2.5C, 1PE2

Learning Objective: Given a DOS Jobstream, revise the JCL to take advantage of all applicable DOS-E (ADAS, DYNAM-T) benefits.

Ref: Instruction Notes

532-74F-1262	Exam and Review/Critique	Type
Hours -	6	2C, 4E

Learning Objective: N/A

Ref: N/A

Annex K Total Hours Allotted: 26

Annex L - Code DOS Librarian Programs

Terminal Learning Objective: Given all class references, code the jobstreams to display a library directory, display and/or punch a library entry and perform maintenance or any or all DOS libraries. The jobstream must be syntactically correct and must meet stated requirements.

532-74F-1265	DOS Library Service Routines	Type
Hours -	5.5	3C, 2.5PE2

Learning Objective: Given all class references, code a jobstream using SSERV, CSERV, RSERV and/or DSERV without syntax errors and with all stated requirements met.

Ref: OS Handout
System Control & Service
DOS Job Control, Shelley/Cashman

532-74F-1266	DOS Library Maintenance Routines	Type
Hours -	6.5	3.5C, 3PE2

Learning Objective: Given all class references, code a jobstream to perform any MAINT function on any library. The jobstream must be without syntax errors and must meet stated requirements.

Ref: System Control & Service
DOS Job Control, Shelley/Cashman

Annex L Total Hours Allotted: 12

Annex M - Code DOS Utilities and Sorts

Terminal Learning Objective: Given all class references and job requirements, code a jobstream using file to file utilities and the SORT program. The jobstream must be syntactically correct and must produce output that meets stated requirements.

532-74F-1270	DOS File to File Utilities	Type
Hours -	6	3C, 3PE2

Learning Objective: Given all class references and job requirements, code a jobstream to transfer data from one storage media to another. The jobstream must meet requirements and be syntactically correct.

Ref: DOS & TOS Utilities
OS Handout

532-74F-1270	DOS SORT/MERGE	Type
Hours -	9	3C, 6PE2

Learning Objective: Given all class references and job requirements, code a jobstream to sort records into a given sequence. The jobstream must result in sorted output and must be syntactically correct.

Ref: DOS Tape and Dist SORT/MERGE Program

532-74F-1271	CSC Utilities	Type
Hours	1	1C

Learning Objective: Given the CSC Utilities Manual index, find the desired utility within 3 minutes.

Ref: CSC Utilities Manual

532-74F-1272	Utilities Review & Exam	Type
Hours -	6	2C, 4E

Learning Objective: N/A

Ref: N/A

Annex M Total Hours Allotted: 22

18

Annex N - Report Systems Problems to USACSC

532-74F-1275	Complete an Incident Report	Type
Hours -	1.5	1.5PI

Learning Objective: Given a scenario and blank forms, complete an Incident Report (USACSC Form 53) with all required information.

Ref: DAP Interface With USACSC

532-74F-1276	Complete a System Change Request	Type
Hours -	1.5	1.5PI

Learning Objective: Given a scenario and blank forms, complete a Systems Change Request with all required information.

Ref: DPA Interface With USACSC

532-74F-1277	CSC/DPA Interface Exam	Type
Hours -	1	1E

Learning Objective: N/A

Ref: N/A

Annex N Total Hours Allotted: 4

29

Annex O - Produce Systems Documentation

Terminal Learning Objective: Given sample DOD documentation standards and program specifications, compile an ADP documentation packet in the proper format and with all required information.

532-74F-1280	Programmer's Documentation	Type
Hours -	6	3C, 3PE2

Learning Objective: Given sample DOD standards and applicable specifications, compile program documentation in the right format and with all required information.

Ref: DOD Documentation Standards

532-74F-1281	Operations Documentation	Type
Hours -	5	3C, 2PE2

Learning Objective: Given sample DOD standards and applicable specifications, compile operations documentation in the right format and with all required information.

Ref: DOD Documentation Standards

532-74F-1282	Documentation Review and Exam	Type
Hours -	4	2C, 2E

Learning Objective: N/A

Ref: N/A

Annex O Total Hours Allotted: 15

20

BIN D-024

DATA REPRESENTATION

PART I

BINARY NUMBER SYSTEM

21

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quite environment that is conducive to strong concentration.

PROCEDURES THAT SHOULD BE FOLLOWED:

Read the frame very carefully.
In some cases the frame should be read twice.
Most of the frames require a response on the part of the student.
Write down the answer that you feel is correct before you turn the page to the answer.
The answer will always be found at the top of the next page.
If your answer is correct continue to the next frame.
If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

22

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Binary Number System. Upon completion of the text, you will be able to:

1. Convert a Binary number to a Decimal number.
2. Convert a Decimal number to a Binary number.
3. Convert a Binary Coded Decimal to a Decimal number.
4. Convert a Decimal number to a Binary Coded Decimal.
5. Perform Binary arithmetic functions.

33

The number system taught in grade school is the Decimal Number System. In this number system ten symbols are used to represent digits.

What are the ten symbols used to indicate digits in the Decimal Number System?

24

ANS: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9

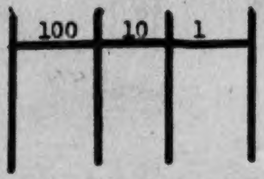
I hope you have not forgotten the Decimal Number System.

The BASE of a number system is the number of symbols used to indicate digits.

What is the BASE of the Decimal Number System?

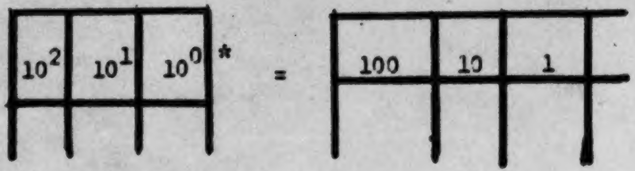
ANS. Since ten symbols are used to represent the digits 0 through 9, the BASE of the Decimal Number System is TEN.

The positional values for three positional columns of the Decimal Number System are:



The positional values of any number system are determined by taking the BASE of the number system and raising it to progressive powers substantially.

Progressive Powers



*Any number raised to the zero power is 1.

Indicate the positional values for four columns of the Decimal Number System.

26

ANS.

1000	100	10	1

or

10^3	10^2	10^1	10^0

The Decimal Number 25 represented with the positional value of columns.

10	1
2	5

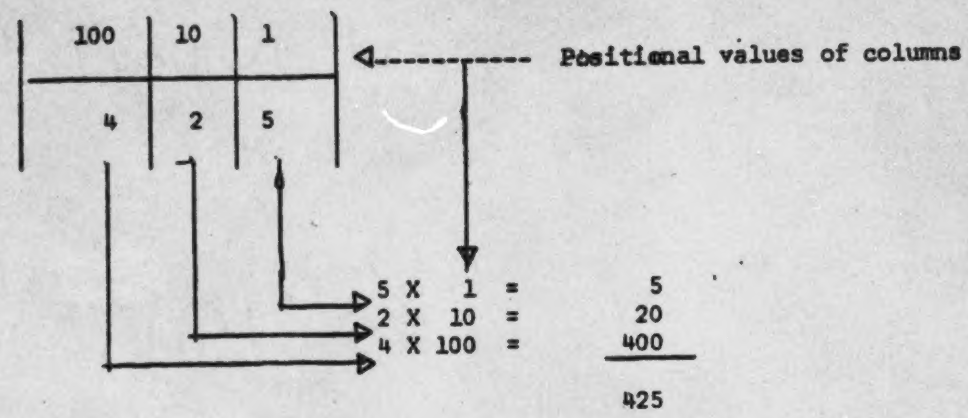
Indicate the positional values over the Decimal Number 425.

ANS.

100	10	1
4	2	5

By using a combination of positional values in the columns, a number can be represented. To determine the value of the number represented, the digit indicated in a column is multiplied by the positional value of the column; then, the values of the digits are added. For example:

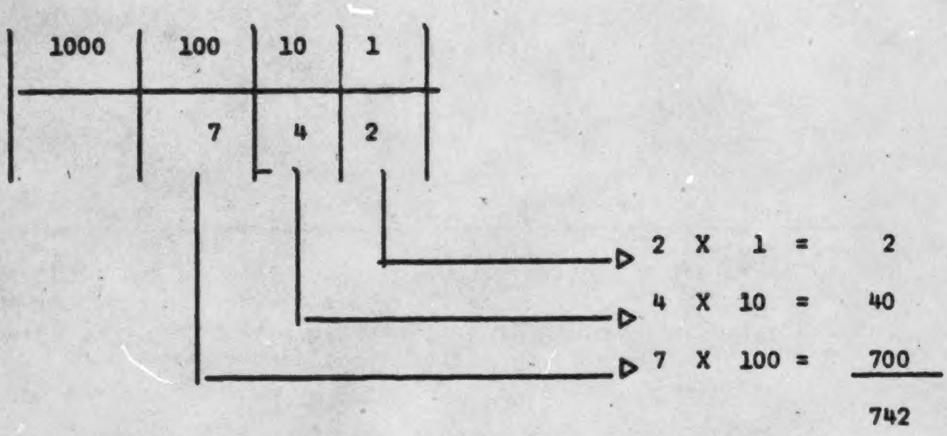
The Decimal Number 425.


100
10
1
4
2
5
Positional values of columns
5 X 1 = 5
2 X 10 = 20
4 X 100 = 400
425

Indicate how the value of Decimal Number 742 is determined.

28

ANS.


1000
100
10
1
7
4
2
2 X 1 = 2
4 X 10 = 40
7 X 100 = 700
742

The largest number that can be indicated in one column of any number system will be the BASE of the number system minus 1.

What is the largest number that can be indicated in one column of the Decimal Number System?

ANS. 9 BASE of Decimal Numbering System = 10
Minus 1 = $-\underline{1}$

9

or

1000	100	10	1
			9

What is the largest number that can be represented in two positional columns?

30.

ANS. 99 or

10	1
9	9

The electronic circuitry used in data processing systems function in binary states. This means that the circuitry can indicate only two possible states, either an "on" or "off" condition. This characteristic made it necessary to develop a number system that could be used by a computer. This number system is called the Binary Number System.

In this number system only the symbols 0 and 1 are used.

What is the BASE of the Binary Number System?

31

ANS: 2

The BASE of a number system is the number of symbols used to indicate digits. In the Binary Number System only two symbols, 0 and 1, are used.

If the BASE of the Binary System is TWO, the positional values for four columns are:

2^3	2^2	2^1	2^0

or

8	4	2	1

What are the positional values for seven colums of the Binary Number System?

32

ANS:

2^6	2^5	2^4	2^3	2^2	2^1	2^0

or

64	32	16	8	4	2	1

What is the largest digit that can be indicated in one positional column in the Binary Number System?

ANS: 1 BASE of Binary Number System = 2
Minus 1 = - 1

1

As in any number system, using a combination of positional values in the columns a number can be constructed or represented. A combination of 1's and 0's in this number system can represent a decimal number.

Method of converting Binary number to Decimal number.

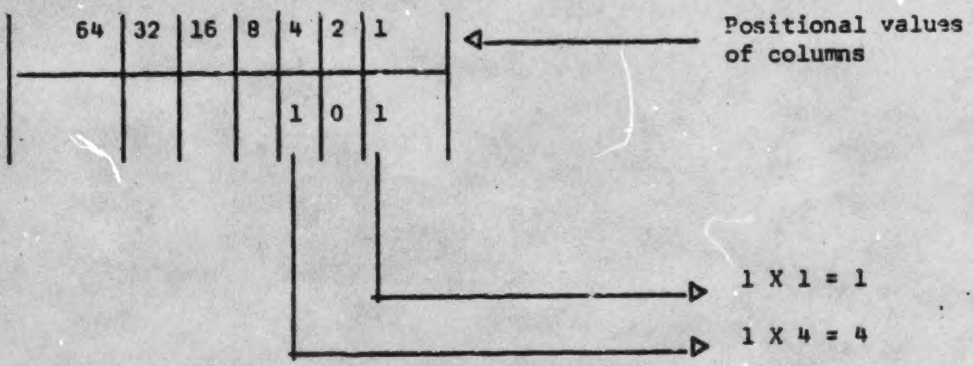
Example is of Binary number 101 converted to a Decimal number.

1. Assign positional values of Binary Number System above the Binary number.

64	32	16	8	4	2	1
				1	0	1

"1" = "on" condition
"0" = "off" condition

2. Multiply digit indicated in each column by its positional value.



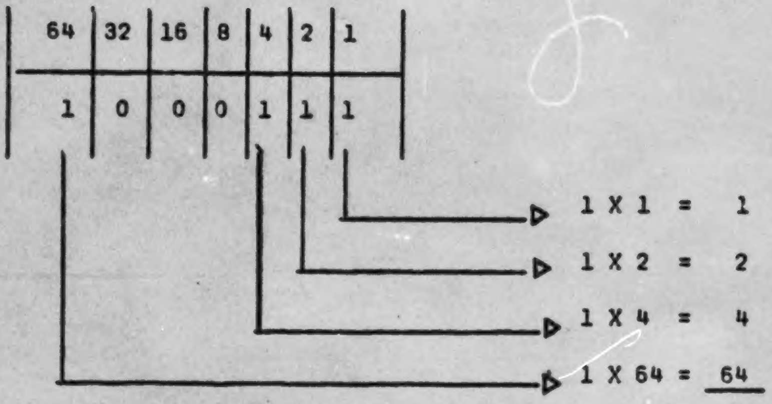
3. Add positional values of columns.

$$\begin{array}{r} 1 \times 1 = 1 \\ 1 \times 4 = \underline{4} \\ 5 \end{array}$$

From this conversion example we have learned that the Binary number 101 is equivalent to the Decimal number 5. Dots under the Binary number indicate positional columns. The columns are represented internally in a computer with a bit that can be turned "on" or "off". So the dot may also represent a bit.

Convert the Binary number 1000111 to a Decimal number.

ANS: 71


64
32
16
8
4
2
1
1
0
0
0
1
1
1
1 X 1 = 1
1 X 2 = 2
1 X 4 = 4
1 X 64 = 64

71 (Decimal)
(Number)

Convert Binary 1010101 to a Decimal Number.

36

ANS: 85

64	32	16	8	4	2	1
1	0	1	0	1	0	1

1 X 1 = 1
1 X 4 = 4
1 X 16 = 16
1 X 64 = 64

85 (Decimal)
(Number)

Convert Binary 0111001 to a Decimal Number.

ANS: 57

64	32	16	8	4	2	1
0	1	1	1	0	0	1

1 X 1 = 1
1 X 8 = 8
1 X 16 = 16
1 X 32 = 32

57 (Decimal)
(Number)

TEST

Convert following Binary numbers to Decimal. Each dot (.) under the Binary number indicates a positional column.

	BINARY		DECIMAL
a.	101 •••	=	________
b.	10001 •••••	=	________
c.	1001 ••••	=	________
d.	111001 ••••••	=	________

38

Method to convert Decimal number to Binary number.

EXAMPLE: Decimal number 91 converted to a Binary number.

1. Assign positional values of the number system the Decimal number is to be converted to. The highest positional value must be larger than the Decimal number to be converted.

Decimal number
to be converted

91

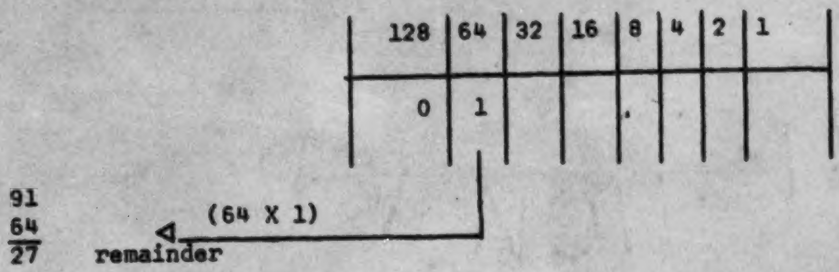
128	64	32	16	8	4	2	1

2. Indicate in the column the number of times the positional value will go into the Decimal number. "0" indicates positional value will not go into Decimal number. Begin with highest positional value.

91

128	64	32	16	8	4	2	1
0	1						

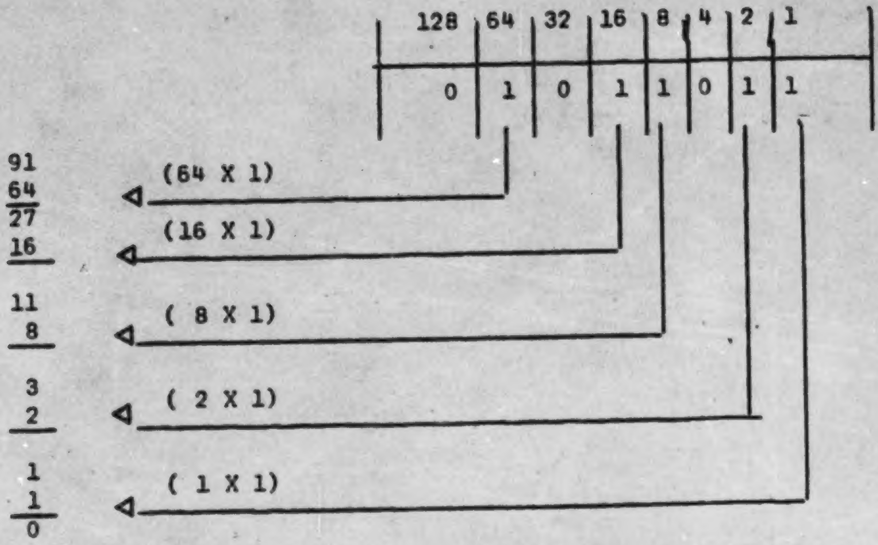
3. Multiply the positional value by the digit indicated in the column; subtract result from Decimal number.



128	64	32	16	8	4	2	1
0	1						

91
64
27 remainder
(64 X 1)

4. Indicate the number of times the next positional value will go into the remainder.



128	64	32	16	8	4	2	1
0	1	0	1	1	0	1	1

91
64
27
16
11
8
3
2
1
1
0
(64 X 1)
(16 X 1)
(8 X 1)
(2 X 1)
(1 X 1)

The Decimal number 91 is equivalent to the Binary number 1011011.

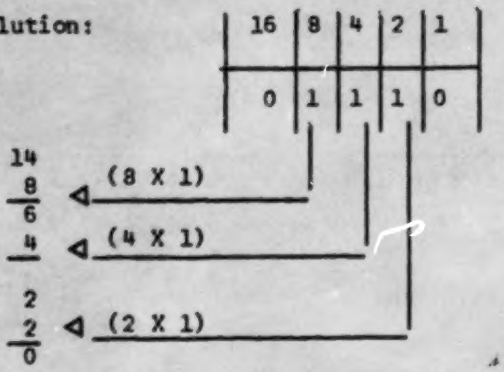
Convert the Decimal number 14 to a Binary number.

50

40.

ANS: 01110

Solution:


16
8
4
2
1
0
1
1
1
0
14
8
6
4
2
2
0
(8 X 1)
(4 X 1)
(2 X 1)

Convert the Decimal number 72 to a Binary number.

41

ANS: 01001000

Convert the Decimal number 45 to a Binary number.

52

42

ANS: 0101101
....

TEST

Convert following Decimal numbers to Binary.

	Decimal		Binary
a.	4	=	....
b.	15	=	.....
c.	26	=	......
d.	63	=	....

BINARY CODED DECIMAL

A Binary number like 01000011011101100101 may be difficult to interpret because of the long string of 1's and 0's. To convert this Binary number to Decimal would involve assigning positional values to 20 positional columns, then adding their positional values. This would take a lot of time! For readability purposes and to save time, the Binary Coded Decimal was designed to easily convert a Binary number to a Decimal number. In Binary Coded Decimal (BCD) four bits are used to indicate one Decimal number. The positional values of these four bits are the same as the first four bits in the Binary Number System.

What are the positional values of the four bits in BCD?

44

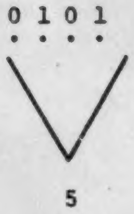
ANS:

8	4	2	1

The Decimal number 5 represented with four bits in Binary Coded Decimal (BCD).

8	4	2	1
0	1	0	1

thur

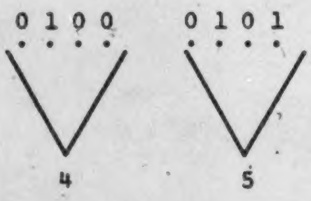


The Decimal number 45 represented in BCD:

8	4	2	1
0	1	0	0

8	4	2	1
0	1	0	1

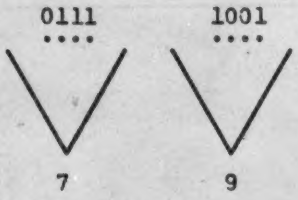
thus,



In BCD only the digits 0 through 9 can be indicated in one group of four bits.

Convert the Decimal number 79 to BCD.

ANS: 0111 1001


0111
1001
7
9

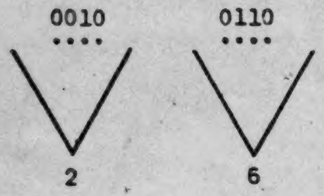
Convert the Decimal number 458 to BCD.

46

ANS: 0100 0101 1000

Convert BCD 0010 0110 to a Decimal number.

ANS: 26


0010
0110
2
6

Convert BCD 0111 0011 0100 1001 to a Decimal number.

ANS: 7349

0111 ••••	0011 ••••	0100 ••••	1001 ••••
7	3	4	9

TEST

1. Convert Binary Coded Decimal (BCD) to Decimal.

	8421	8421	8421	100	10	1
a.			0011 ••••			•
b.	••••	0010 ••••	0010 ••••		•	•
c.	••••	0101 ••••	0100 ••••		•	•
d.	0111 ••••	0110 ••••	1000 ••••	•	•	•

2. Convert Decimal to BCD.

	100	10	1	8421	8421	8421
a.			7			••••
b.		1	5		••••	••••
c.		7	6		••••	••••
d.	4	2	3	••••	••••	••••

BINARY ADDITION

Rules applied when performing Binary addition:

a. 0 + 0 = 0

b. 1 + 0 or 0 + 1 = 1

d. 1 + 1 = 0 and carry 1 to next column to left

d. 1 + 1 + 1 = 1 and carry 1 to next column to left

Addition problem: $\begin{array}{r} 010 \\ +\ \underline{011} \end{array}$

1 carried

$$\begin{array}{r} 1 \\ 010 \\ +\ \underline{011} \\ 101 \end{array} \qquad \text{same as} \qquad \begin{array}{r} 2 \\ +\ \underline{3} \\ 5 \end{array}$$

Add: $\begin{array}{r} 101 \\ +\ \underline{101} \end{array}$

50
ANS: 1010

Solution:

$$\begin{array}{r} {}^{1\,1} \\ 101 \\ +\ \underline{101} \\ 1010 \end{array}$$

Add:

$$\begin{array}{r} 001 \\ +\ \underline{111} \end{array}$$

51

ANS: 1000

Add:

```
  0110
+ 1100
```

52

ANS: 10010

Add: 0110101
+ 1110100

ANS: 10101001

TEST

Perform Binary addition

```
a.    101              b.    1000
    + 001                  + 1001
      ---                  ------
      ...                  .....

c.   1011              d.   01110
   + 1001                   11011
   ------                   -----
    .....                  ......
```

BINARY SUBTRACTION

The method by which subtraction is performed internally in a computer is Subtraction by complementation. The complement of a number is the amount that must be added to a certain number to have it indicate the largest number that can be indicated in one positional column. In Binary the complement of 1 is 0, the complement of 0 is 1 and, really, all we are doing is reversing the binary notation of the subtrahend (the number that is to be subtracted from another).

```
Subtraction problem:        101
                          - 011
```

a. Place the proper sign bit before each binary number; 0 = plus (+), 1 = minus (-). A plus sign is applied to minuend (top number) when minuend does not have plus or minus sign.

```
                  ------ Sign bits
       101  }  | 0 | 101
     - 011  }  | 1 | 011
```

b. Complement subtrahend and add. Do not complement minuend or sign bits.

```
  | 0 |  101                      | 0 |  101
  | 1 |  011  (Subtrahend)  -->   | 1 |  100
                                 1| 0 |  001
```

Anything to left of sign bit is dropped or truncated. Thus 1| 0 | 001 becomes | 0 | 001.

c. Add 1 to result just obtained.

```
  | 0 | 001
  |   |   1
  | 0 | 010   <-- Sign bit is 0 or +.  Thus answer is + 2.
```

```
Subtract:     100
            - 010
```

56

```
ANS: | 0 | 010        Solution:                     sign bit

               100 }   | 0 | 100
             - 010 }   | 1 | 101
                       ---------
                     1 | 0 | 001    =   | 0 | 001
                                        |   | + 1
                                        ---------
                                        | 0 | 010    or   +2
```

Subtract:

```
  1000
- 0111
```

ANS: | 0 | 0001

```
Subtract:    10110
           - 01101
```

58

ANS: 9 01001

TEST

Subtract:

a. 0111
- 0101

b. 1000
- 0100

c. 1001
- 0110

d. 10001
- 01010

DATA REPRESENTATION

PART II

OCTAL NUMBER SYSTEM

60

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.
In some cases the frame should be read twice.
Most of the frames require a response on the part of the student.
Write down the answer that you feel is correct before you turn the page to the answer.
The answer will always be found at the top of the next page.
If your answer is correct continue to the next frame.
If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

71

61

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Octal Number System. Upon completion of the text, you will be able to:

1. Convert an Octal number to a Decimal number.
2. Convert a Decimal number to an Octal number.
3. Convert Binary Coded Octal to a Decimal number.
4. Convert a Decimal number to Binary Coded Octal.
5. Perform Octal arithmetic functions.

62

OCTAL NUMBER SYSTEM

In the Octal Number System EIGHT symbols are used to represent its digits.

What is the BASE of the Octal Number System?

73

63

ANS: The BASE of the Octal Number System is EIGHT. This is because eight symbols are used to indicate digits in this number system.

Since the BASE of the Octal Number System is EIGHT, the positional values of the columns will be the BASE of the number system raised to progressive powers sequentially.

What is the positional value of the first column in the Octal Number System?

64

ANS: The first column will always be the BASE of the number system raised to the zero power. Any number to the zero power is one. Thus; First column = 8^0 or 1, which is indicated as

8^0

or

1

The positional values of three columns of the Octal number system are:

8^2	8^1	8^0

or

64	8	1

What are the positional values for four columns of the Octal number system?

ANS:

8^3	8^2	8^1	8^0

or

512	64	8	1

The largest number that can be indicated in one column of any number system will be the BASE of the number system minus 1.

What is the largest number that can be indicated in one column of the Octal Number System?

66

ANS: 7

$$\begin{array}{lcr} \text{BASE of Octal Number System} & = & 8 \\ \text{Minus 1} & = & -\underline{1} \\ & & 7 \end{array}$$

If the largest number that can be indicated in each column is 7, what are the symbols to indicate digits in the Octal Number System?

ANS: 0, 1, 2, 3, 4, 5, 6, and 7

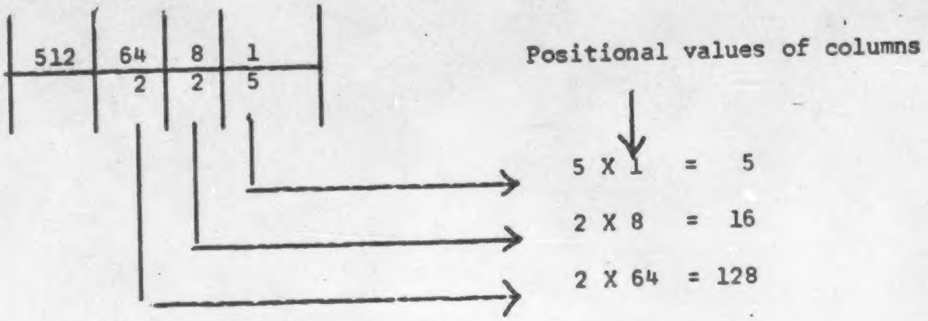
Method for converting OCTAL number system to DECIMAL number.

Example of the OCTAL number 225 converted to a DECIMAL number.

1. Assign positional values of OCTAL number system above the OCTAL number.

512	64	8	1
	2	2	5

2. Multiply digit indicated in each column by its positional value.


512
64
8
1
2
2
5
Positional values of columns

5 X 1 = 5

2 X 8 = 16

2 X 64 = 128

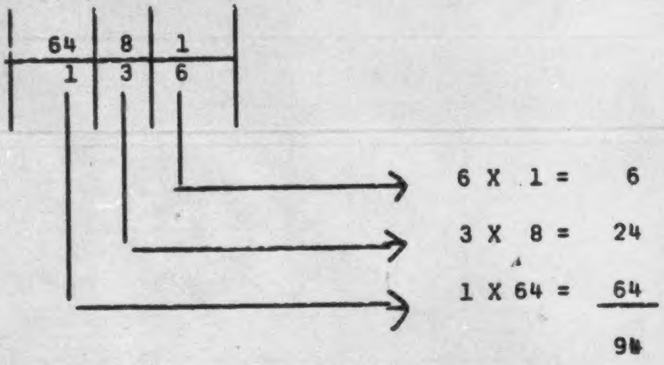
3. Add positional values of columns

$$\begin{array}{rcr} 5 \times 1 & = & 5 \\ 2 \times 8 & = & 16 \\ 2 \times 64 & = & \underline{128} \\ & & 149 \end{array}$$

Thus, the OCTAL number 225 is equivalent to the DECIMAL number 149.

Convert the OCTAL number 136 to a DECIMAL number.

ANS: DECIMAL 94 Solution:



64	8	1
1	3	6

6 X 1 = 6

3 X 8 = 24

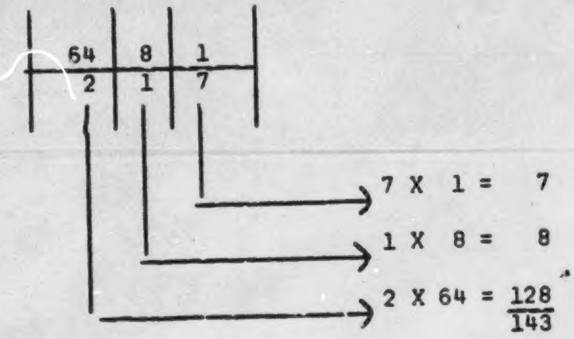
1 X 64 = 64

94

Convert the Octal number 217 to a Decimal number.

.70

ANS: DECIMAL 143 Solution:


64
8
1
2
1
7
7 X 1 = 7
1 X 8 = 8
2 X 64 = 128
143

TEST

Convert OCTAL number to DECIMAL number

	8	1	10	1
a.	1	2	.	.
b.	1	7	.	.
c.	3	2	.	.
d.	7	6	.	.

STEPS TO CONVERT DECIMAL NUMBER TO OCTAL NUMBER.

The Decimal number 89 converted to an Octal number.

1. Assign positional values of Octal Number System. The highest positional value must be larger than Decimal Number to be converted.

Highest Positional Value

512	64	8	1

Decimal Number to be Converted

↓

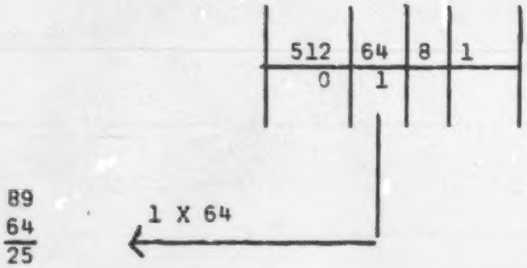
89

2. Indicate in the column the number of times the positional value will go into the Decimal number. "0" indicates positional value will not go into Decimal number. (Begin with highest positional value)

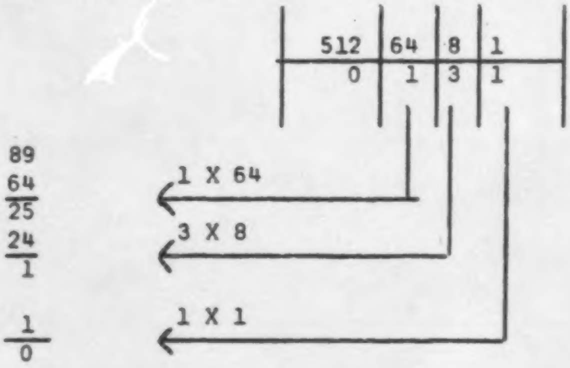
512	64	8	1
0			

89

3. If positional value in the column will go into the Decimal number, multiply the positional value that goes into the Decimal number by the digit indicated in the column; then, subtract result from the Decimal number.


512
64
8
1
0
1
89
64
25
1 X 64

4. Indicate in the following columns the number of times the positional values will go into remainder until remainder is zero.


512
64
8
1
0
1
3
1
89
64
25
24
1
1
0
1 X 64
3 X 8
1 X 1

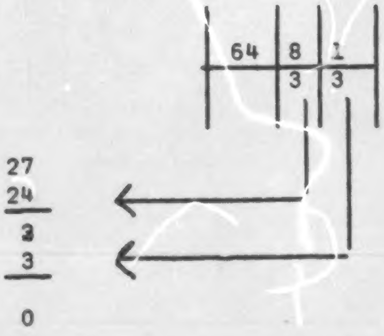
Thus, the Decimal number 89 is equivalent to the Octal number 131.

Convert the Decimal number 27 to an Octal number.

73

ANS: Octal 33

Solution:

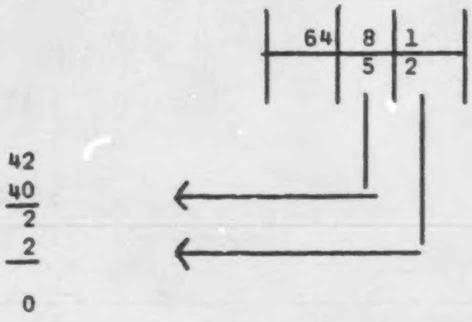

64
8
1
3
3
27
24
3
3
0

Convert the Decimal number 42 to an Octal number.

74

ANS: Octal 52

Solution:


64
8
1
5
2
42
40
2
2
0

Convert the Decimal number 205 to an Octal number.

ANS: OCTAL 315 Solution:

512	64	8	1
0	3	1	5

205
192 ← (3 X 64)
13
8 ← (1 X 8)
5
5 ← (5 X 1)
0

TEST

Convert Decimal to Octal

	10	1	8	1
a.		6		.
b.	1	8	.	.
c.	4	6	.	.
d.	6	3	.	.

76

/

BINARY CODED OCTAL (BCO)

In Binary Coded Decimal one decimal number was represented with 1's and 0's in a group of four bits (four columns). In Binary Coded Octal three bits are used to represent one Octal number with 1's and 0's. The positioned values of these three bits are the same as the first three bits in the Binary Numbering System.

What are the positional values of the three bits in Binary Coded Octal?

ANS:

4	2	1

The OCTAL number 6 is represented with three bits in Binary Coded Octal:

4	2	1
1	1	0

thus, $\underbrace{\underset{\cdot}{1}\ \underset{\cdot}{1}\ \underset{\cdot}{0}}_{6}$

The OCTAL number 36 represented in Binary Coded Octal.

4	2	1
0	1	1

4	2	1
1	1	0

thus, $\underbrace{\underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{1}}_{3}\quad\underbrace{\underset{\cdot}{1}\ \underset{\cdot}{1}\ \underset{\cdot}{0}}_{6}$

In Binary Coded Octal only the digits 0 through 7 can be indicated in one group of three bits.

Convert the Binary Coded Octal number $\underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{0}\quad\underset{\cdot}{1}\ \underset{\cdot}{0}\ \underset{\cdot}{0}$ to an Octal number.

78

ANS: 24 Solution: $\underbrace{\underset{\cdot}{0}\,\underset{\cdot}{1}\,\underset{\cdot}{0}}_{2}\quad\underbrace{\underset{\cdot}{1}\,\underset{\cdot}{0}\,\underset{\cdot}{0}}_{4}$

Convert the Binary Coded Octal number $\underset{\cdot}{1}\,\underset{\cdot}{0}\,\underset{\cdot}{1}\quad\underset{\cdot}{0}\,\underset{\cdot}{0}\,\underset{\cdot}{1}$ to an Octal number.

ANS: 51 Solution: $\underbrace{1\ 0\ 1}_{5}\ \underbrace{0\ 0\ 1}_{1}$

Convert OCTAL number 43 to Binary Coded Octal.

80:

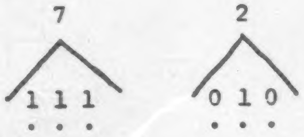
ANS: $\underset{\cdot\ \cdot\ \cdot}{1\ 0\ 0}\quad\underset{\cdot\ \cdot\ \cdot}{0\ 1\ 1}$

Solution $\overbrace{\underset{\cdot\ \cdot\ \cdot}{1\ 0\ 0}}^{4}\quad\overbrace{\underset{\cdot\ \cdot\ \cdot}{0\ 1\ 1}}^{3}$

Convert OCTAL number 72 to Binary Coded Octal.

ANS: 1 1 1   0 1 0

Solution



TEST

1. Convert Binary Coded Octal to Octal number.

	4 2 1	4 2 1	8 1	
a.	0 0 1	0 0 1	. .	(Octal number)
b.	0 0 1	1 0 1	. .	(Octal number)
c.	1 0 1	1 1 0	. .	(Octal number)
d.	1 1 1	1 0 0	. .	(Octal number)

2. Convert Octal number to Binary Coded Octal.

	64 8 1	4 2 1	4 2 1	4 2 1
a.	5			. . .
b.	2 3		. . .	. . .
c.	5 2		. . .	. . .
d.	1 2 7	. . .	. . .	. . .

OCTAL ADDITION:

$$\begin{array}{r} 43 \\ +\ \underline{56} \end{array}$$

1. Add one column at a time.

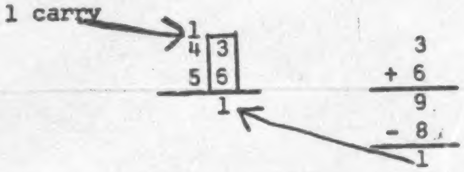
$$\begin{array}{r} 4 \\ +\ 5 \\ \hline \end{array}\!\!\boxed{\begin{array}{c} 3 \\ 6 \end{array}} \qquad \begin{array}{r} 3 \\ 6 \\ \hline 9 \end{array}$$

2. If result is 7 or less indicate result in column. But if result is more than 7, subtract result by 8 (the base of the Octal number system).

$$\begin{array}{r} 4 \\ +\ 5 \\ \hline \end{array}\!\!\boxed{\begin{array}{c} 3 \\ 6 \end{array}} \qquad \begin{array}{r} 3 \\ +6 \\ \hline 9 \\ -\ 8 \\ \hline 1 \end{array} \leftarrow$$

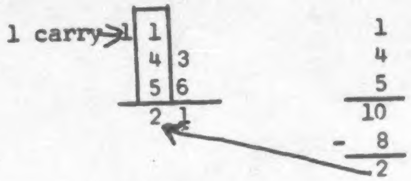
3. Indicate remainder in the column added and carry 1 to next column.

First column:

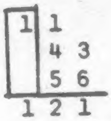

1 carry
1
4 3
5 6
1
3
+ 6
9
- 8
1

4. Add following columns in same manner (include 1 carry).

Second column:


1 carry
1
4 3
5 6
2 1
1
4
5
10
- 8
2

Third column:



Thus, the result of the OCTAL addition is 121.

Perform OCTAL addition:

$$\begin{array}{r} 2 \\ +\ 7 \\ \hline \end{array}$$

84
ANS: 11

Perform OCTAL addition:

```
  42
+ 55
----
```

95

ANS: 117

TEST

Perform OCTAL addition:

a. $\begin{array}{r} 2 \\ +\ 5 \\ \hline \end{array}$

b. $\begin{array}{r} 12 \\ +\ 7 \\ \hline \end{array}$

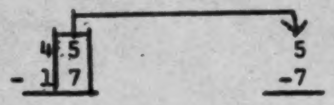
c. $\begin{array}{r} 52 \\ +66 \\ \hline \end{array}$

d. $\begin{array}{r} 64 \\ +31 \\ \hline \end{array}$

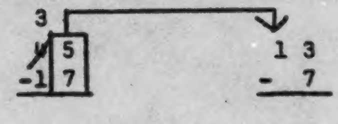
OCTAL SUBTRACTION

$$\begin{array}{r} 45 \\ -\ 17 \\ \hline \end{array}$$

1. Subtract one column at a time.



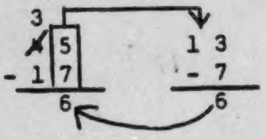
2. If minuend (number to be subtracted from) is smaller than subtrahend, borrow the base (8) from next column.



(5 + 8 (Base))

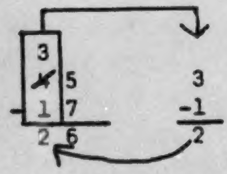
3. Indicate result in the column subtracted

First Column:



4. Subtract following column in same manner

Second Column:



Thus, the result of the OCTAL subtraction is 26.

Perform OCTAL subtraction:

$$\begin{array}{r} 34 \\ -\ 15 \\ \hline \end{array}$$

88

ANS: 17

Perform OCTAL subtraction:

$$\begin{array}{r} 7\ 3 \\ -\ 3\ 5 \\ \hline \end{array}$$

99

89

ANS: 36

TEST

Perform OCTAL subtraction:

```
     5 2                     6 4
a.  -1 4                 b. -2 5
    ----                    ----


     3 7                     6 6
c.  -1 7                 d. -5 7
    ----                    ----
```

100

DATA REPRESENTATION

PART III

HEXADECIMAL NUMBER SYSTEM

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.
In some cases the frame should be read twice.
Most of the frames require a response on the part of the student.
Write down the answer that you feel is correct before you turn the page to the answer.
The answer will always be found at the top of the next page.
If your answer is correct continue to the next frame.
If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

92

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Hexadecimal Number System. Upon completion of the text, you will be able to:

1. Convert a Hexadecimal number to a Decimal number.
2. Convert a Decimal number to a Hexadecimal number.
3. Convert Binary Coded Hexadecimal to a Decimal number.
4. Convert a Decimal number to Binary Coded Hexadecimal.
5. Perform Hexadecimal arithmetic functions.

103

HEXADECIMAL NUMBER SYSTEM

In the Hexadecimal Number System SIXTEEN symbols are used to represent it's digits.

What is the BASE of the Hexadecimal Number System?

94

ANS: The BASE of the HEXADECIMAL number system is SIXTEEN. This is because sixteen symbols are used to represent digits in this number system.

Since the BASE of the HEXADECIMAL number system is SIXTEEN, the positional values of the columns will be the BASE of the number system raised to progressive powers sequentially.

What is the positional value of the first column in HEXADECIMAL number system?

ANS:

16^0

or

1

The first column will always be the BASE of the number system raised to the zero power. Any number to the zero power is one. Thus first column =-16^0 or 1.

The positional values of three columns of the HEXADECIMAL number system are:

16^2	16^1	16^0

or

256	16	1

What are the positional values for four columns of the HEXADECIMAL number system?

96

ANS:

16^3	16^2	16^1	16^0

or

4096	256	16	1

The largest number that can be indicated in one column of any numbering system will be the BASE of the numbering system minus 1.

What is the largest number that can be indicated in one column of the HEXADECIMAL number system?

ANS: 15 Solution:

BASE of Hexadecimal Number System = 16
Minus 1 = - 1
15

In the HEXADECIMAL number system alphabetic letters are used to indicate 10 through 15.

10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

If the largest digit that can be indicated in each column is F (which is 15), what are the symbols that are used to indicate digits in the Hexadecimal number system?

ANS: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

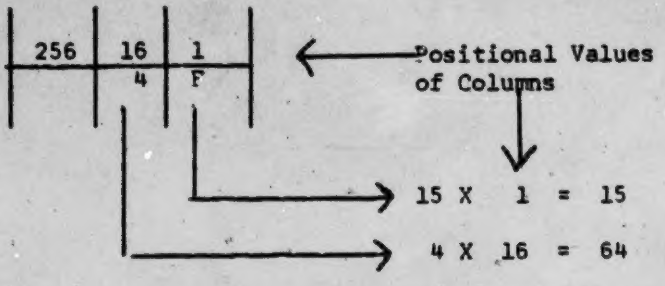
Method for converting HEXADECIMAL number system to DECIMAL number system.

Example of the HEXADECIMAL number 4F converted to a DECIMAL number.

1. Assign positional values of HEXADECIMAL number system above the HEXADECIMAL number.

256	16	1
	4	F

2. Multiply digit indicated in each column by its positional value.


256
16
1
4
F
Positional Values of Columns
15 X 1 = 15
4 X 16 = 64

3. Add positional values of columns

```
15 X  1 = 15
 4 X 16 = 64
          79
```

Thus, the HEXADECIMAL number 4F is equivalent to the DECIMAL number 79.

Convert HEXADECIMAL 1B to a DECIMAL number.

100

ANS: 27 **Solution:**

256	16	1
	1	B

11 X 1 = 11

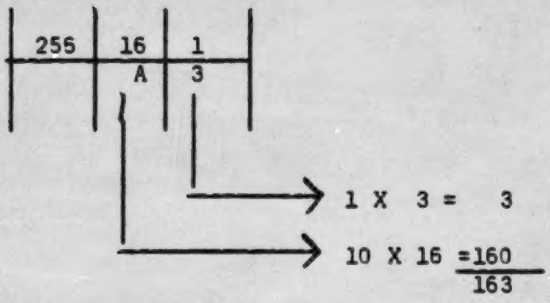
1 X 16 = 16

27

Convert HEXADECIMAL A3 to a DECIMAL number?

101

ANS: 163 Solution:



255
16
1
A
3
1 X 3 = 3
10 X 16 = 160
163

TEST

Convert HEXADECIMAL to DECIMAL.

	256	16	1	100	10	1
a.			E		.	.
b.		A	9	.	.	.
c.		F	3	.	.	.
d.	1	C	6	.	.	.

Method for converting DECIMAL number system to HEXADECIMAL number system.

Example of the DECIMAL number 167 converted to a HEXADECIMAL number.

1. Assign positional values of HEXADECIMAL number system. The highest positional value must be larger than DECIMAL number to be converted.

Decimal number
to be converted
↓
167

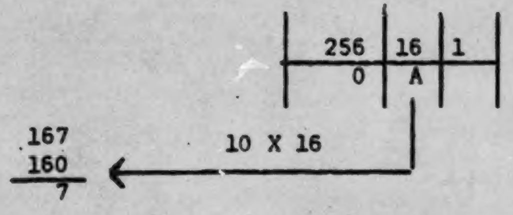
256	16	1

2. Indicate in the column the number of times the positional value will go into the DECIMAL number. "0" indicates positional value will not go into DECIMAL number (Begin with highest positional value.)

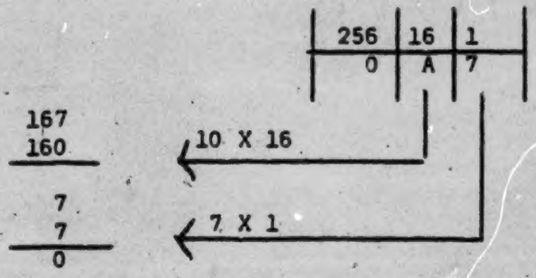
167

256	16	1
0		

3. If positional value in the column will go into the DECIMAL number, multiply the positional value that goes into the DECIMAL number by the digit indicated in the column; then, subtract result from the DECIMAL number.


256
16
1
0
A
10 X 16
167
160
7

4. Indicate in the following columns the number of times the positional values will go into remainder until remainder is zero.

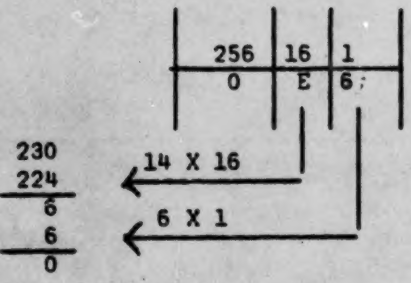

256
16
1
0
A
7
167
160
10 X 16
7
7
0
7 X 1

Thus the DECIMAL number 167 is equivalent to the HEXADECIMAL number A7.

Convert the DECIMAL number 230 to a HEXADECIMAL number.

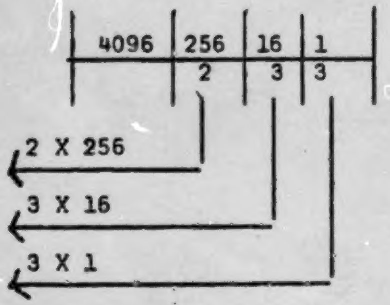
104

ANS: E6 **Solution:**


256
16
1
0
E
6
230
224
6
6
0
14 X 16
6 X 1

Convert the DECIMAL number 563 to a HEXADECIMAL number.

ANS: 233 Solution:


4096
256
16
1
2
3
3
2 X 256
3 X 16
3 X 1

$$\begin{array}{r}563\\ \underline{512}\\ 51\\ \underline{48}\\ 3\\ \underline{3}\\ 0\end{array}$$

TEST

Convert Decimal to Hexadecimal

	1000	100	10	1	256	16	1.
a.			1	3		.	.
b.		1	7	5	.	.	.
c.		9	4	0	.	.	.
d.	2	6	1	4	.	.	.

106

BINARY CODED HEXADECIMAL (BCH)

In Binary Coded Decimal, one decimal number was represented with 1's and 0's in a group of four bits (four columns). In Binary Coded Hexadecimal, again four bits are used to represent one Hexadecimal number with 1's and 0's. The positional values of these four bits are the same as the first four bits in the Binary Numbering System.

What are the positional values of the four bits in the Binary Coded Hexadecimal?

ANS:

8	4	2	1

The Hexadecimal number A (10) represented in Binary Coded Hexadecimal (with four bits):

8	4	2	1
1	0	1	0

thus, $\underbrace{\underset{\cdot}{1}\ \underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{0}}_{A}$

The Hexadecimal number 6A represented in Binary Coded Hexadecimal.

8	4	2	1
0	1	1	0

8	4	2	1
1	0	1	0

thus, $\underbrace{\underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{1}\ \underset{\cdot}{0}}_{6}\qquad\underbrace{\underset{\cdot}{1}\ \underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{0}}_{A}$

In Binary Coded Hexadecimal the digits 0 through F (15) can be represented in one group of four bits.

Convert the Binary Coded Hexadecimal number $\underset{\cdot}{1}\ \underset{\cdot}{0}\ \underset{\cdot}{1}\ \underset{\cdot}{1}\quad \underset{\cdot}{0}\ \underset{\cdot}{0}\ \underset{\cdot}{0}\ \underset{\cdot}{1}$ to a Hexadecimal number.

108.

ANS: B1 Solution: $\underbrace{\underline{1}\,\underline{0}\,\underline{1}\,\underline{1}}_{B}\quad\underbrace{\underline{0}\,\underline{0}\,\underline{0}\,\underline{1}}_{1}$

Convert the Binary Coded Hexadecimal number $\underline{0}\,\underline{0}\,\underline{1}\,\underline{1}\;\;\underline{1}\,\underline{1}\,\underline{0}\,\underline{0}$ to a Hexadecimal number.

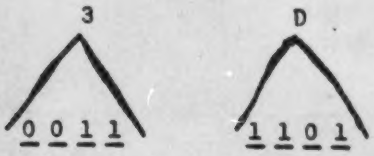
ANS: 3C **Solution:** $\underbrace{\underline{0}\ \underline{0}\ \underline{1}\ \underline{1}}_{3}\quad\underbrace{\underline{1}\ \underline{1}\ \underline{0}\ \underline{0}}_{C}$

Convert Hexadecimal number 3D to Binary Coded Hexadecimal.

110

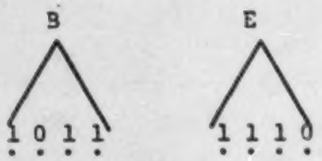
ANS: 0 0 1 1 1 1 0 1

Solution:


3
D
0 0 1 1
1 1 0 1

Convert Hexadecimal number BE to Binary Coded Hexadecimal.

ANS: 1 0 1 1 1 1 1 0 Solution:



TEST

1. Convert Binary Coded Hexadecimal to Hexadecimal.

	8 4 2 1	8 4 2 1	8 4 2 1	256	16	1
a.			0 1 1 1			.
b.		0 1 1 0	1 1 0 1		.	.
c.		1 1 0 0	0 1 1 0		.	.
d.	0 0 1 0	1 0 1 1	1 1 1 1		.	.

2. Convert Hexadecimal to Binary Coded Hexadecimal.

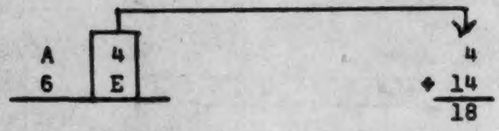
	256	16	1	8 4 2 1	8 4 2 1	8 4 2 1
a.			C			. . . .
b.		E	4		. . . .	. . . .
c.		9	A		. . . .	. . . .
d.	2	B	B	. . . .	. . . .	. . . .

HEXADECIMAL ADDITION:

A4
6E

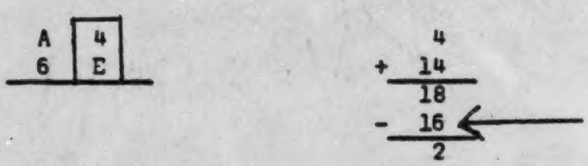
1. Add one column at a time.

First Column:



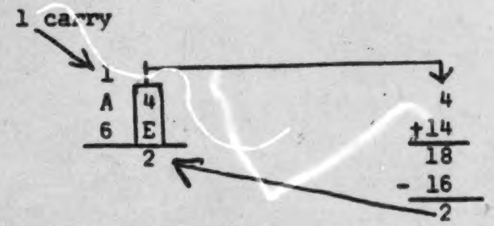
2. If result is F (15) or less indicate result in column (in Hexadecimal). But if result is more than F, subtract result by 16 (the base of the Hexadecimal number system).

First Column:



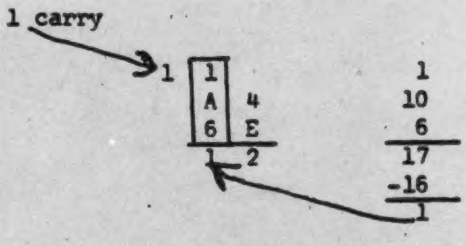
3. Indicate remainder in the column added and carry 1 to next column:

First Column:

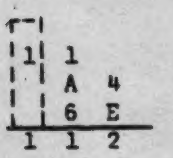

1 carry
1
A 4
6 E
2
4
+14
18
- 16
2

4. Add following columns in same manner (include 1 carry).

Second Column:


1 carry
1 1
A 4
6 E
1 2
1
10
6
17
-16
1

Third Column:



Thus, the result of the Hexadecimal addition is HEXADECIMAL 112.

Perform Hexadecimal addition:

$$\begin{array}{r} 8 \\ +\ 3 \\ \hline \end{array}$$

114

ANS: B (11)

Perform Hexadecimal addition: $\begin{array}{r} B\,7 \\ \underline{1\,3} \end{array}$ 125

115

ANS: C A

Perform Hexadecimal addition:

```
  C D
+ B E
```

126

ANS: 18B

TEST

Perform Hexadecimal addition:

a. $\begin{array}{r} 8 \\ +\ 5 \\ \hline \end{array}$

b. $\begin{array}{r} A6 \\ +\ 45 \\ \hline \end{array}$

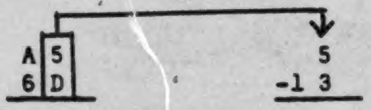
c. $\begin{array}{r} B4 \\ +\ 6E \\ \hline \end{array}$

d. $\begin{array}{r} D\ E \\ C\ F \\ \hline \end{array}$

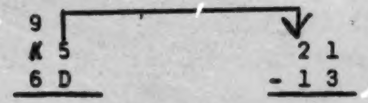
HEXADECIMAL SUBTRACTION

$$\begin{array}{r} A5 \\ +\ 6D \\ \hline \end{array}$$

1. Subtract one column at a time.



2. If minuend (number to be subtracted from) is smaller than subtrahend, borrow the base (16) from next column.



[5 + 16 (BASE)]

3. Indicate result in the column subtracted.

First Column:

$$\begin{array}{r} 9 \\ A\ 5 \\ 6\ D \\ \hline 8 \end{array} \qquad\qquad \begin{array}{r} 2\ 1 \\ -\ 1\ 3 \\ \hline 8 \end{array}$$

4. Subtract following column in same manner.

Second Column:

$$\begin{array}{r} 9 \\ \cancel{A}\ 5 \\ 6\ D \\ \hline 3\ 8 \end{array} \qquad\qquad \begin{array}{r} 9 \\ -\ 6 \\ \hline 3 \end{array}$$

Thus, the result of Hexadecimal subtraction is 38.

Perform Hexadecimal subtraction:

$$\begin{array}{r} 9\ 4 \\ -\ 7\ C \\ \hline \end{array}$$

ANS: 18

Perform Hexadecimal subtraction

```
  C 3
- 6 B
```

120

ANS: 58

TEST

Perform Hexadecimal subtraction:

$$\begin{array}{r} 4\ 1 \\ -2\ B \\ \hline \end{array}$$ a.

$$\begin{array}{r} A\ 2 \\ -8\ 7 \\ \hline \end{array}$$ b.

$$\begin{array}{r} C\ 4 \\ -A\ 1 \\ \hline \end{array}$$ c.

$$\begin{array}{r} D\ C \\ -A\ C \\ \hline \end{array}$$ d.

DATA REPRESENTATION

PART IV

BCDIC and EBCDIC *

---

* An understanding of the Hollerith Code is a prerequisite for this part of Programmed Instruction Text. P.I. TEXT on Punched Cards is TC 14-71-50PT.

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.
In some cases the frame should be read twice.
Most of the frames require a response on the part of the student.
Write down the answer that you feel is correct before you turn the page to the answer.
The answer will always be found at the top of the next page.
If your answer is correct continue to the next frame.
If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the various codes used to represent alphabetic, special, and numeric data internally within a computer system. Upon completion of the text you will be able to:

1. Interpret the Binary Coded Decimal Interchange Code.
2. Interpret the Extended Binary Coded Decimal Interchange Code.

BINARY CODED DECIMAL INTERCHANGE CODE

In Binary Coded Decimal, four bits were used to represent one decimal digit and only digits could be represented in this code. By extending this four bit field by two more bits we can represent alphabetic characters as well as numbers. This second generation six bit field is known as the BINARY CODED DECIMAL INTERCHANGE CODE (BCDIC).

```
B A 8 4 2 1
• • • • • •
```

Name the two punches required on a punched card to represent characters.

ANS: Zone punch and digit punch

The zone and digit bits of the Binary Coded Decimal Interchange Code:

$$\overbrace{\text{BA}}^{\text{Zone Bits}} \qquad \overbrace{8421}^{\text{Digit Bits}}$$

Zone bits used to represent zone punches.

(Punch) Zone		(Bits) BA
12	=	11
11	=	10
0	=	01

What letters can be represented in the Hollerith Code with:
   a. 12 zone punch?
   b. 11 zone punch?
   c. 0 zone punch?

126

ANS:

a. A through I
b. J through R
c. S through Z

Representation of characters:

CHARACTER		ZONE PUNCH	DIGIT PUNCH	ZONE BITS BA	DIGIT BITS 8421
A	=	12,	1	11	0001
J	=	11,	1	10	0001
S	=	0,	2	01	0010
9	=	(not used),	9	00	1001

What are the Hollerith Code punches used to represent the letter "B"?

137

127

ANS: "12" zone punch, "2" digit punch.

Indicate the letter "B" in the Binary Coded Decimal Interchange Code.

128

ANS:

B	A	8	4	2	1
1	1	0	0	1	0

Indicate the letter "M" in BCDIC.

139

ANS:

B	A	8	4	2	1
1	0	0	1	0	0
•	•	•	•	•	•

Indicate the letter "V" in BCDIC.

130

ANS:

B	A	8	4	2	1
0	1	0	1	0	1

Indicate the number "6" in BCDIC.

141

ANS:

B	A	8	4	2	1
0	0	0	1	1	0

TEST

Indicate character in BCDIC.

	CHARACTER	BA 8 4 2 1
a.	U	. . . . . .
b.	D	. . . . . .
c.	L	. . . . . .
d.	4	. . . . . .

Special characters used BCDIC will not be discussed in the Programmed Instruction Text. IBM System 360 Reference Data Card provides the bit configuration of special characters in BCDIC.

PARITY CHECK

Whenever data is transfered from one device to another in a computer system, there is always a chance that a bit condition ("on" or "off") may be lost or gained. This of course, would change the character representation. An additional check bit position has been added to the six bit field (BCDIC) to make what is called a parity check so we can determine when a bit has been lost or gained:

check bit

C B A 8 4 2 1
• • • • • • •

When making an odd parity check, the number of "1" bits under the BA8421 are counted. If the number of "1" bits is even, another "1" bit is placed under the check bit (C) to make the entire number of 7 bits odd. For even parity the entire number of 7 bits is even.

The letter "C" with

1. Odd Parity: C B A 8 4 2 1
   1 1 1 0 0 1 1
2. Even Parity: C B A 8 4 2 1
   0 1 1 0 0 1 1

Place the proper check bit condition ("1" or "0") for the letter "B". Check for odd parity:

C B A 8 4 2 1
_ 1 1 0 0 1 0
• • • • • • •

134

ANS:

```
C B A 8 4 2 1
0 1 1 0 0 1 0
_ • • • • • •
```

Place the proper check bit condition for the letter "E". Check for odd parity.

```
C B A 8 4 2 1
- 1 1 0 1 0 1
  • • • • • •
```

```
ANS:        C B A 8 4 2 1
            1 1 1 0 1 0 1
            - • • • • • •
```

Place the proper check bit condition for the letter "A". Check for even parity.

```
C B A 8 4 2 1
- 1 1 0 0 0 1
• • • • • • •
```

136

ANS:

```
C B A 8 4 2 1
1 1 1 0 0 0 1
- . .
```

Place the proper check bit condition for the letter "K". Check for even parity.

```
  C B A 8 4 2 1
.   1 0 0 0 1 0
  - . . . . . .
```

147

ANS:

C	B	A	8	4	2	1
0	1	0	0	0	1	0

TEST

Place proper check bit condition:

1. Odd Parity

	C	B	A	8	4	2	1
a.	-	1	1	0	1	0	0
b.	-	1	0	0	1	1	1
c.	-	1	1	1	0	0	1
d.	-	0	1	0	0	1	1

2. Even Parity

	C	B	A	8	4	2	1
a.	-	1	0	0	1	0	1
b.	-	0	1	0	1	1	1
c.	-	1	1	1	0	0	0
d.	-	1	0	0	1	0	0

EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE

In the Binary Coded Decimal Interchange Code, alphabetic characters could be represented as either capital letters (upper case) or small letters (lower case), but not in both cases. By extending the bit, we can represent both upper and lower cases of letters. The new third generation eight bit code is known as the Extended Binary Coded Decimal Interchange Code (EBCDIC).

CASE BITS	ZONE BITS	DIGIT BITS
84	21	8421
••	••	••••

Case bits $\overset{8\ 4}{\overline{1\ 1}}$ indicate capital letters (upper case).
Case bits 1 0 indicate small letters (lower case).
Hexadecimal numbers are indicated by using 1's for both case & zone bits.

Representation of Characters:

HOLLERITH CODE

CHARACTER		ZONE PUNCH	DIGIT PUNCH	CASE BITS 84	ZONE BITS 21	DIGIT BITS 8421
A	=	12,	1	11	00	0001
a	=	(not applicable)		10	00	0001
J	=	11,	1	11	01	0001
j	=	(not applicable)		10	01	0001
S	=	0	2	11	10	0010
s	=	(not applicable)		10	10	0010
9	=	(not used),	9	11	11	1001

Note that the "on" and "off" condition of the zone bits in EBCDIC is the reverse of the "on" and "off" condition of the zone bits in BCDIC.

Indicate the following in EBCDIC.
- a. Capital "B"
- b. Small "b"

140.

ANS:

	8421	8421		
a.	1100	0010	=	B
b.	1000	0010	=	b

Indicate the following in EBCDIC.
a. Capital "M"
b. Small "m"

151

ANS:		8421	8421	
	a.	1101	0100	= "M"
	b.	1001	0100	= "m"

Indicate following in EBCDIC.

a. Capital "V"
b. Number "7"

		8421	8421	
ANS:	a.	1110	0101	= "V" *
	b.	1111	0111	= "7"

*Remember that "V" is 0 zone punch, 5 digit punch.

TEST

Indicate character in EBCDIC.

CHARACTER	8421	8421
a. 4	....	....
b. D	....	....
c. t (small)	....	....
d. 4	....	....

Special characters used in EBCDIC will not be discussed in this Programmed Instruction. IBM System 360 Reference Data Card provides the bit configuration of special characters in EBCDIC.

A parity check can also be made in EBCDIC by providing an additional check bit. With the exception that there are more bits in EBCDIC than in BCDIC, the method of checking for parity is the same.

The smallest unit of data in a computer is a Bit. Eight bits form the next smallest unit of data, a Byte. In EBCDIC, you've seen that we can use one byte to represent either an alphabetic character, one hexadecimal number, or one special character.

Examples:

BYTE ZONED DIGIT		BYTE ZONE DIGIT		BYTE ZONE DIGIT	
1111 1001 F 9	= 9	1100 0001 C 1	= A	0101 1011 5 B	= $

For representing only numeric data, EBCDIC is redundant and wasteful of storage space in the computer. This is because for numeric data in EBCDIC, the zone half of each byte is a hexadecimal F (1111). The major exception to this is the usage of plus (+) and minus (-) signs, which are shown in the zone portion of the right most byte of an EBCDIC number. To show a positive number the F is changed to a C, and to show a negative number the F is changed to a D.

Examples:

F 7 F 1 F 6 F 9 = 7169 (This is assumed positive, although not shown)

F 7 F 1 F 6 C 9 = +7169

F 7 F 1 F 6 D 9 = -7169

Convert the following numbers to EBCDIC format.

a. 943

b. - 943

c. + 349

d. - 8767

ANS: a. F9 F4 F3 c. F3 F4 C9

b. F9 F4 D3 d. F8 F7 F6 D7

As you can see, the F zone is repetitiva and wastes storage space. To eliminate this waste, the computer provides the "packed Decimal" or "Binary Coded Decimal" representation. Once again "C" indicates a POSITIVE (+) number, and the "D", a NEGATIVE (-) number. The difference is that each byte contains 2 decimal digits, with the sign shown in the "DIGIT" portion of the rightmost byte in the number.

Example:

Decimal number + 63264

Represented in:
EBCDIC F6 F3 F2 F6 C4
Packed Decimal 63 26 4C

Note that instead of using 5 bytes, we are now able to use only 3 bytes, thus a saving of 2 bytes.

If the number you are packing does not completely fill the area you've allocated for it, zero's (0) are used as fill on the left.

Example:

F8 F3 F2 D1 EBCDIC

-8321 08 32 1D Packed

Indicate the decimal number +421 in EBCDIC format and in packed format.

146

ANS: F 4 F 2 C 1 EBCDIC

4 2 1 C Packed

In "Packing", the proper method for calculating how many bytes will be necessary to accomodate a number is to first count the digits in the number. If it is an even number, add two (2) to it. If it's odd, add one (1). Then divide this number by two. The result will be the number of bytes needed.

Example:

+459 =	3	Digits (odd)	+ 6324 =	4	Digits (even)
	1	Add 1		2	Add 2
	4	Divide by 2		6	Divide by 2
	2	# of bytes		3	# of bytes
	4 5 9 C			0 6 3 2 4 C	

How many bytes will be needed to pack the number?

a. 8762391
b. 67821467821431

ANS: A. = 4

B. = 8

You have seen now how alphabetic characters, number, and special symbols are represented in EBCDIC. You have further seen how we can save storage space by using the "Packed Decimal" feature in representing numbers. Proper knowledge of this is necessary in gaining a complete understanding of computer systems.

SPECIAL TEXT

ST 18-150

DRAFT

BASIC COBOL PROGRAMING

Prepared By

UNITED STATES ARMY INSTITUTE OF ADMINISTRATION
Fort Benjamin Harrison, Indiana — 46216

TABLE OF CONTENTS

Page

INTRODUCTION 3

COBOL LANGUAGE CONSTRUCTION 3
COBOL CHARACTER SET 3
CHARACTERS USED IN WORDS 4
PUNCTUATION RULES 4
CHARACTERS USED IN ARITHMETIC EXPRESSIONS 5
MARGINS AND THE COBOL CODING SHEET 5
WORD 5
COBOL WORDS 6

THE COBOL LANGUAGE STRUCTURE 8

IDENTIFICATION DIVISION 9
PROGRAM-ID 9
AUTHOR 9
DATE-COMPILED 9
REMARKS 9

ENVIRONMENT DIVISION 11
SPECIAL-NAMES 12
SELECT 13

DATA DIVISION 17
FD 17
RECORD DESCRIPTION 21

WORKING-STORAGE SECTION 24
77 LEVEL 24
USAGE 24
VALUE 24
SYNCHRONIZED 24
JUSTIFIED 26
COMPUTATIONAL 26
EDITED FIELDS 29
ARRAYS 31
REDEFINED 32
88 LEVEL 33

PROCEDURE DIVISON 34

PROCESS VERBS 35
MOVE 35
ADD 40
SUBTRACT 42
MULTIPLY 43
DIVIDE 44
COMPUTE 45

150

Page

INPUT-OUTPUT 46
  READ 46
  WRITE 47
  DISPLAY 50
  EXHIBIT 50
  OPEN 51
  CLOSE 52

LOGIC FLOW 52
  IF 52
  PERFORM 59
  GO TO 62
  STOP 63
  ON 64

DEBUGGING 65
  TRACE 65
  RESET 65

TABLE HANDLING 65
  SUBSCRIPTING 66
  INDEXING 68

DOS JCL
OS JCL

AMERICAN NATIONAL STANDARD COBOL
for
IBM System/360 Disk Operating System & Operating System

This manual is designed as a student introductory text to 'ANSI' COBOL. Since this is an introductory text, information contained herein may contain over-simplified explanations of the COBOL syntax. More explicit information may be obtained from IBM System/360 Disk Operating System American National Standard COBOL, File Number S360-24, GC28-6394 or IBM System/360 Operating System American National Standard COBOL GC28-6396 (IBM Systems Reference Library, DOS). Information contained herein is extracted mostly from these publications. Army standards were taken from U.S. Army Computer Systems Command Manual 18-1-1, dated 15 May 1975.

The following extract from Government Printing Office Form Number 1965-0795689 is presented for the information and guidance of the user:

"Any organization interested in reproducing the COBOL report and specifications in whole or in part, using ideas taken from this report as the basis for an instruction manual or for any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention 'COBOL' in acknowledgement of the source, but need not quote this entire section."

"COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations." "No warranty, expressed or implied, is made by any contributor or by the COBOL Committee as to the accuracy and functioning of the programing system and language. Moreover, no responsibility is assumed by any contributor, or by the committee in connection therewith." Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages." "The authors and copyright holders of the copyrighted material used herein have specifically authorized the use of this material in whole or

> FLOW-MATIC (Trademark of Sperry Rand Corporation), Programing for the UNIVAC (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial translater, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260, copyrighted 1960 by Minneapolis-Honeywell.

in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programing manuals or similar publications."

Information in this text represents the language promulgated by the Conference On Data Systems Languages (CODASYL) (the USA standard of the language is American National Standard COBOL, X3.23-1968 (formerly known as USA Standard COBOL, as approved by the American National Standards Institute (ANSI), unless specifically stated otherwise. COBOL is a problem oriented language which is designed to be highly self-documenting. The structure of the language helps document the program as it is written.

INTRODUCTION

COBOL LANGUAGE CONSTRUCTION. COBOL is based on English. It uses English-type words and certain syntax rules derived from English. However, because it is a computer language, it is much more precise than English. The programer must, therefore, learn the rules that govern COBOL and follow them exactly.

The basic unit of COBOL is the word. This may be either a COBOL reserved word or a programer-defined word. Reserved words have a specific syntactical meaning to the COBOL compiler and must be spelled exactly as shown. Programer-defined words are assigned by the user to such items as data-names (a name made up by the programer to refer to a specific piece of data in memory). Reserved words and programer-defined words are combined by the programer into clauses and statements.

Clauses and statements must be formed following the specific syntactical rules of COBOL. A clause or a statement specifies only one action to be performed, one condition to be analyzed, or one description of data. Clauses and statements can be combined into sentences.

Sentences may be simple (one statement or combination of clauses). Sentences can be combined into paragraphs, which are named units of logically related sentences, and paragraphs can be further combined into named sections.

Both paragraphs and sections can be referred to as procedures, and their names can be referred to as procedure names.

Procedures (sections and paragraphs) are combined into divisions. Divisions are joined into a COBOL program. Each program has exactly four divisions.

COBOL CHARACTER SET. The following symbols may be used in writing COBOL:

SYMBOL	MEANING	SYMBOL	MEANING
0,1,...9	digits	,	comma
A,B,...Z	letters	;	semicolon
+	plus sign	.	period
-	minus/hyphen sign	'	quotation mark (ANSI shows a quote as ")
*	asterisk	(	left parenthesis
/	slash	)	right parenthesis
=	equal sign	>	greater than symbol
$	currency sign	<	less than symbol
			space (no printed character)

Characters Used in Words

The characters used in words in a COBOL source program are:
0 thru 9
A thru Z
- (hyphen)

A word is composed of a combination of not more than 30 characters chosen from the character set for words. The word cannot begin or end with a hyphen.

The following characters are used for punctuation:

	space	.	period
,	comma	)	right parenthesis
;	semicolon	(	left parenthesis
'	quotation mark		

PUNCTUATION RULES

The following general rules of punctuation apply in writing COBOL source programs:

1. A period, semicolon, or comma, when used, must not be preceded by a space, but must be followed by a space.
2. A left parenthesis must not be followed immediately by a space; a right parenthesis must not be preceded by a space.
3. At least one space must appear between two successive words and/or parenthetical expressions and/or literals. Two or more successive spaces are treated as a single space, except within non-numeric literals.
4. An arithmetic operator or an equal sign must always be preceded by a space and followed by a space.
5. A comma may be used as a separator between successive operands of a statement. An operand of a statement is shown in a format as a lower case word.
6. A comma or semicolon may be used to separate a series of clauses and is used for ease of reading as in written English.
7. A semicolon may be used to separate a series of statements. Example: ADD A TO B; SUBTRACT B FROM C.

Characters Used in Arithmetic Expressions:

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation
(	left parenthesis
)	right parenthesis
=	replacement operator

NOTE: When any punctuation mark is indicated in a format (in this publication), it is required in the program.

MARGINS AND THE COBOL CODING SHEET

When writing a COBOL program, the programer is concerned with 2 margins: margin A and margin B. Margin A encompasses columns 8 through 11 of a coding sheet. Margin B encompasses columns 12 through 72.

Margin A is used to identify the different divisions, paragraphs, sections, or special items of a COBOL program. Entries in Margin A may continue through column 72, if necessary, and are terminated by a period. Each statement or clause should be on a single line (card). Margin A statements do not have to begin in column 8, but <u>must</u> begin before column 12.

Army Standard: All entries that can begin in Margin A must begin in column 8.

Margin B contains all the information the programer wishes to include within a defined division, paragraph, section, or special items of the program. Margin B statements do not <u>have</u> to begin in column 12, but may be written anywhere from column 12 through column 72 (see below).

EXAMPLE:

```
SPECIAL-NAMES.
    C01 IS CHAN 1.
```

The word SPECIAL-NAMES begins in margin A.
The words C01 IS CHAN 1 begins in margin B.

WORD

A WORD is composed of not more than 30 characters from the COBOL character set for word formation. The space (blank) is not an allowable character in a word; the space is a word separator. A word is terminated by a space, a period, a right parenthesis, a comma, or a semicolon.

Throughout this publication, basic formats are prescribed for various elements of COBOL. These generalized descriptions are intended to guide the programer in writing his own statements. They are presented in a uniform system of notation, explained in the following paragraphs. Although it is not part of COBOL, this notation is useful in describing COBOL.

COBOL WORDS

1. All words printed entirely in capital letters are reserved words. These words have preassigned meanings in COBOL. In all formats, words in capital letters represent an actual occurrence of those words. If any such word is incorrectly spelled, it will not be recognized as a reserved word and will probably cause an error in the program. Reserved words cannot be used as procedure or data names.

2. All underlined reserved words are required unless the portion of the format containing them is itself optional. These are key words. If any such word is missing or is incorrectly spelled, it is considered an error in the program. Reserved words not underlined may be included or omitted at the option of the programer. These words are used only for the sake of readability; they are called optional words, and when used, must be correctly spelled.

3. The characters + - = when appearing in formats, although not underlined, are required when such formats are used.

4. All punctuation and other special characters (except those symbols cited in the following paragraphs) represent the actual occurrence of those characters. Punctuation is essential where it is shown. Additional punctuation can be inserted according to the rules for punctuation specified in this publication.

5. Words that are printed in lower-case letters represent information to be supplied by the programer. No entry in lower case letters may be omitted, unless it is part of an optional section or phrase which is omitted.

6. In order to facilitate references to them in text, some lower-case words are followed by a hyphen and a digit or letter. This modification does not change the syntactical definition of the word.

7. Square brackets ([]) are used to indicate that the enclosed item may be used or omitted depending on the requirements of the particular program. When two or more items are stacked within brackets, one or none of them may occur.

8. Braces ({}) enclosing vertically stacked items indicate that one of the enclosed items is obligatory.

157

9. The ellipsis (...) indicates that the immediately preceding unit may occur once, or any number of times in succession. A unit means either a single lower-case word, or a group of lower-case words and one or more reserved words enclosed in brackets or braces. If a term is enclosed in brackets or braces, the entire unit of which it is a part must be repeated when repetititon is specified.

10. Programer defined words are names chosen by the programer to identify program-unique things such as file names or data names. They should be descriptive and clear to someone not familiar with the program.

Army Standard: Only Army standard abbreviations in AR 18-12 may be used in programer defined words.

THE COBOL LANGUAGE STRUCTURE

There are four divisions in each COBOL program. Each is placed in its logical sequence, each has its necessary logical function in the program, and each uses information developed in the divisions preceding it. The four divisions and their sequences are:

```
IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.
```

IDENTIFICATION DIVISION

The purpose of the IDENTIFICATION DIVISION is to give the who-what-when-why-where-how of the program to a maintenance programmer. It also identifies the program to the operating system.

BASIC FORMAT:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.        program-name.
[AUTHOR.            your name.]
[INSTALLATION.      where the program is written.]
[DATE-WRITTEN.      the date that coding began.]
[DATE-COMPILED.     this entry is completed by the compiler.]
[SECURITY.          the classification of the program.]
[REMARKS.           a brief description of what the program accomplishes.]
```

The first words of any COBOL program are IDENTIFICATION DIVISION. IDENTIFICATION DIVISION must begin in margin A:

```
IDENTIFICATION DIVISION.
```

Note that there is one space between IDENTIFICATION and DIVISION, and that IDENTIFICATION DIVISION terminates with a period.

The only required part of the IDENTIFICATION DIVISION is the PROGRAM-ID clause. Since "program-name" is in lower case letters, the programer must make up a descriptive name by which his program may be identified. All other IDENTIFICATION DIVISION entries help to describe and document your program.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SAMPLE-1.
```

Note that PROGRAM-ID is written in margin A and follows IDENTIFICATION DIVISION which is written in margin A. After PROGRAM-ID, there is at least one space (a period must be followed by one or more spaces). Then the programer writes the name of his program, in this case, SAMPLE-1. The program name may be placed anywhere in margin B. An alternate example, which means exactly the same thing as the first example, is listed below.

EXAMPLE:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                 SAMPLE-1.
```

All other clauses of the IDENTIFICATION DIVISION are optional. They may be included or deleted as desired. Each clause begins with the word AUTHOR, INSTALLATION, DATE-WRITTEN, DATE-COMPILED, SECURITY, or REMARKS. Each is followed by a period, followed by any comment-type entry, then terminated by a period. This entry, called a sentence, may be repeated as many times as desired.

EXAMPLE:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.      PILPMGT
 AUTHOR.          LT HUDGIN, ATSG-D-S.
 INSTALLATION.    USAIA, FORT HARRISON, IN  46216.
 DATE-WRITTEN.    14 AUGUST 1976.
 DATE-COMPILED.   TODAY
 SECURITY.        UNCLASSIFIED.
*REMARKS.         THIS IS A SAMPLE OF A COMPLETE IDENTIFICATION
*    DIVISION.  ALL OF THE ENTRIES ARE INCLUDED IN THIS EXAMPLE,
*    ALTHOUGH ONLY THE PROGRAM-ID PARAGRAPH (WHICH INCLUDES THE
*    PROGRAM-NAME) IS REQUIRED.
```

Army Standards: Although most entries are not required by ANS COBOL, it is a good practice to include them for documentation purposes.

PROGRAM-ID: The PROGRAM-ID will consist of six positions constructed as follows:

Position	Entry
1	a. For a program that is operational, this position will contain a 'P'. b. For a program that is being tested, this position will have a 'T'. c. If the program is under development, enter a 'D' in this position.
2-3	These two positions identify the specific program within a system or subsystem.
4-6	The subsystem's or system's identification code.

AUTHOR: Enter the programer's name and/or the office symbol responsible for the program.

DATE-COMPILED: Some compilers will not properly place the compilation date if entry is blank. To avoid this, enter 'TODAY' in this area.

REMARKS: Some compiliers do not support this paragraph. To avoid errors, enter an asterisk (*) in column 7 of every card in this paragraph. This paragraph should give the reason the program was written, basically how it works, any special processing or techniques used, what all 77 and 01 entries in the WORKING-STORAGE SECTION are used for, and any communications with other programs. Each time a change is made, the maintenance programmer should add an entry giving the date, his name, why the change was made, and what the change encompassed.

ENVIRONMENT DIVISION

The ENVIRONMENT DIVISION is used for the following purposes:

a. To identify the computer system to be used (e.g., IBM 360-30, UNIVAC 1108, etc.).

b. To identify the peripheral devices to be used (e.g., card readers, printers, tape drives, etc.).

c. To give or assign a logical name to each peripheral device (e.g., SYS006, SYS004, etc.).

d. To identify the files to be used by the program.

e. To define the organization (sequential, random, etc.) of each file.

f. To assign each file to a specific peripheral device.

BASIC FORMAT:

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
[SOURCE-COMPUTER. computer-name.]
[OBJECT-COMPUTER. computer-name.]
[SPECIAL-NAMES. [function-name-1 IS condition-name-1]]...
INPUT-OUTPUT SECTION.
FILE-CONTROL.

(DOS)*

$$\begin{array}{l}\underline{\text{SELECT}}\ \text{file-name}\\ \quad\underline{\text{ASSIGN}}\ \text{TO}\end{array}\quad \underline{\text{SYSnnn-}}\left\{\begin{array}{c}\underline{\text{UT}}\\ \underline{\text{UR}}\end{array}\right\}\underline{\text{-}}\left\{\begin{array}{c}\underline{\text{2314}}\\ \underline{\text{1403}}\\ \underline{\text{2400}}\\ \underline{\text{2540R}}\\ \underline{\text{2540P}}\end{array}\right\}\underline{\text{-}}\ \underline{\text{S}}.$$

(OS)*

$$\begin{array}{l}\underline{\text{SELECT}}\ \text{file-name}\\ \quad\underline{\text{ASSIGN}}\ \text{TO}\end{array}\quad \left\{\begin{array}{c}\underline{\text{UT}}\\ \underline{\text{UR}}\end{array}\right\}\underline{\text{-}}\left[\begin{array}{c}\underline{\text{2314}}\\ \underline{\text{1403}}\\ \underline{\text{2400}}\\ \underline{\text{2540R}}\\ \underline{\text{2540P}}\end{array}\ \underline{\text{-}}\right]\underline{\text{S}}\ \underline{\text{-}}\ \text{external-name.}$$

* See note on next page.

* This paragraph by definition is machine and operating system dependent. We have provided examples of its coding for the two most common operating systems in the Army. No attempt is made to limit this instruction only to IBM hardware or software. Each manufacturer has programing manuals available to explain their own implementation of this paragraph. If device independence is desired when using OS, the device model number must be omitted from the system name and the device class must be UT even if the file is on a unit record device when the program is executed.

There are 2 sections of the ENVIRONMENT DIVISION: The CONFIGURATION SECTION and the INPUT-OUTPUT SECTION. The CONFIGURATION SECTION specifies which computer is to be used to compile the program (the SOURCE-COMPUTER), which computer will be used to execute the program once it is compiled (the OBJECT-COMPUTER), and associates names in the COBOL program with special tests on peripheral devices used in the program.

EXAMPLE:

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-360-G30.
OBJECT-COMPUTER.  IBM-360-G30.
```

In this example, both the source and object computers are the same. The descriptions should be of the smallest computer that the program can be compiled or executed on.

SPECIAL-NAMES is part of the CONFIGURATION SECTION and is used if the programer wishes to skip to a particular line on the printer page (for example, to the top of the next page). The IBM 1403 printer has a carriage control tape with 12 channels, each of which may have a hole punched in it to indicate where to stop the paper if that particular channel is searched. The carriage control tape used for student jobs has a punch indicating the top of page is on channel 1 and the bottom of the page is on channel 09. The programer must specify a name which will be associated with the hole punched in the channel if he wishes to skip (slew) to that point on the page. The names to be associated with the channels are specified in the SPECIAL-NAMES clause:

Army Standard: Both the SOURCE-COMPUTER and OBJECT-COMPUTER are required entries.

```
SPECIAL-NAMES.
    Cnn IS mnemonic-name(...).
```

"C" means channel; nn is a two-digit number indicating which channel of the twelve possible is to be associated with the name. The mnemonic name is the name that the programer gives to the channel. It will be used in the PROCEDURE DIVISION skipping to the channel is desired.

EXAMPLE:

```
SPECIAL-NAMES.
    C01 IS CHAN1
    C09 IS CHAN9.
```

Here the word CHAN1 will later be recognized as meaning skip to channel 1 of the carriage control tape on the printer. CHAN9 will mean skip to channel 09.

Army Standards: The only special names that may be used and their purpose are:

Special Name	Purpose
CHAN1	Indicates the first print line page (Channel 1).
CHAN9	Indicates the last print line of a page (Channel 9).

Channel 1 on the Army standard channel tape is line 6 on the printer page. Channel 9 is line 66. On standard 11" long printer paper printed six lines per inch that leaves one inch margins at the top and bottom.

The FILE-CONTROL paragraph of the INPUT-OUTPUT SECTION is used to tell the system how many files will be used, what type of peripheral devices the files are on, the names to be used to refer to each file, the organization of the files, and where the files are located. Each file used is specified at least once by the use of a SELECT clause and its parts.

To tell the system that a file will be used, write the word SELECT (following the INPUT-OUTPUT SECTION and FILE-CONTROL cards) in margin B. Then devise a name for the file. This name is written after SELECT.

EXAMPLE 1:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MASTER-FILE
```

EXAMPLE 2:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INVENTORY
```

In the above 2 examples, the programer made up the word MASTER-FILE in the first example, and the word INVENTORY in the second. From this point on, these words must be used whenever the programer wishes to do anything to the file (e.g., READ or OPEN). The name chosen for the file cannot be a COBOL reserved word.

After the file name, the programer writes ASSIGN TO followed by a system name. The system name can have two forms: one for DOS and one for OS. A DOS system name for a file consists of the logical unit assignment (system logical numbers), the class of device, the model number of the specific device, and the file organization. An OS system name for a file consists of the class, the file organization, and the external name.

In DOS, the system logical number refers to a specific hardware device assigned to that number by JCL (Job Control Language). It's value is a number from SYS000 to SYS243, depending on the parameters chosen during system generation. DOS and OS JCL are discussed later in this text. In OS, the external name performs the same function as the system logical number in DOS. It is limited to from 1 to 8 characters, must begin with a letter, and contain no special characters.

Class refers to the device's capability to handle one or more than one record length. A punched card, for example, is always the same length; it has 80 characters represented on it at all times. UNIT-RECORD (UR) is the class used to describe this type device. Even though no punches are punched in the card, there are still 80 characters there; the space is a character to the computer. When read, therefore, there must be sufficient memory allocated to hold 80 characters, whether they are spaces or any of the other permissible characters of the HOLLERITH or EBCDIC code. The same is true when punching a card, the card punch must know what character is to be punched in each of the 80 columns. A blank card to be punched means, in effect, punch no holes. A similar concept holds true for the printer which must know what character is to be printed in each of 132 print positions. These devices, the card-reader, card punch, and printer, are known as UNIT-RECORD devices. They always handle fixed-length records. On the other hand, magnetic tape and disk units are not limited to just one record length. These devices conform to the record length specified for each particular job or file, and when accessing information sequentially, they are referred to as UTILITY (UT) devices.

> The letters UR are used to specify UNIT-RECORD; the letters UT are used to specify UTILITY.

After the class is specified, the programer specifies the model number of the device to be used for the file. This is the hardware model number of the peripherals in the machine configuration at the installation.

For an IBM 360-30 or 360-40, which are the most common Army configurations,

> The card reader is specified by 2540R.
> The card punch is specified by 2540P.
> The disk is specified by 2314.
> The printer is specified by 1403.
> The magnetic tape is specified by 2400.

The file organization will always be sequential for UR and UT devices.

> Sequential organization is specified by S.

All of the parts in the system name are separated by hyphens.

EXAMPLE-1 (DOS). A card file (input):

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MASTER-INVENTORY-FILE
        ASSIGN TO SYS006-UR-2540R-S.
```

EXAMPLE-1 (OS).

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MASTER-INVENTORY-FILE
        ASSIGN TO UT-S-MASTER.
```

The DOS example shows that the name of this file is MASTER-INVENTORY-FILE. The system logical number is SYS006; the card reader is a unit-record device, thus UR; the device number for the card reader is 2540R; and S indicates that it is a SEQUENTIAL file.

The OS example has the same file name, but the device model number is omitted, the class is UT, the organization is sequential, and the external name is MASTER.

EXAMPLE-2 (DOS). An output file on a printer:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT LISTING
        ASSIGN TO SYS005-UR-1403-S.
```

EXAMPLE-2 (OS).

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT LISTING
        ASSIGN TO UT-S-PRINTER.
```

The DOS example shows that the name of the file is LISTING; the system logical number is SYS005; UR for UNIT-RECORD; 1403 which is the device number for the printer; and S for SEQUENTIAL file.

The OS example has the same file name, but the device model number is omitted, the class is UT, the organization is sequential, and the external name is PRINTER.

If a program required 2 files, one for input and the other for output, 2 select statements would be required.

EXAMPLE-3 (DOS). A program requiring 2 files--input from disk and output on a printer:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INPUT-FILE
        ASSIGN TO SYS006-UT-2314-S.
    SELECT OUTPUT-FILE
        ASSIGN TO SYS005-UR-1403-S.
```

EXAMPLE-3 (OS).

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INPUT-FILE
        ASSIGN TO UT-S-DISK FILE
    SELECT OUTPUT-FILE
        ASSIGN TO UT-S-PRINTER.
```

Note that there is one select statement for each file in the program. The outputs file are similar to the ones in Example 2. For the input file:

DOS	OS
The name of the file is INPUT-FILE.	The name of the file is INPUT-FILE.
The system logical number is SYS006.	The device class is UT.
UT for a UTILITY device.	The organization is sequential.
2314 which specifies the type of disk drive on our system.	The external name is DISK FILE.
S for SEQUENTIAL file.	

For the output file:

DOS	OS
The name of the file is OUTPUT-FILE.	The name of the file is OUTPUT-FILE.
The system logical name is SYS005.	The device class is UT.
It is a UTILITY device (UT).	The organization is sequential.
The device is an IBM 1403 printer.	The external name is PRINTER.
It is a sequential file (S).	

DATA DIVISION

The purpose of the DATA DIVISION is:

1. To describe each file in detail.
2. Allocate memory required by the record of each file.
3. Allocate all other memory needed by the program for working areas (scratch pad).

This text explains the DATA DIVISION in 3 parts: the FILE SECTION FD (file description), the FILE SECTION record description, and the WORKING-STORAGE SECTION.

```
DATA DIVISION.
FILE SECTION.
FD file-name
                                    [CHARACTERS]
    [BLOCK CONTAINS integer         [RECORDS   ]  ]

    [RECORD CONTAINS integer-1  [TO interger-2   CHARACTERS]]

             {RECORD IS  }  {OMITTED }
     LABEL   {RECORDS ARE}  {STANDARD}

    [DATA    {RECORD IS  }
             {RECORDS ARE}   data-name-1 [data-name-2] ...]
```

Army Standard: When more than one clause is used to describe a file, each clause must begin on a new card beginning in column 12.

The first part of the DATA DIVISION which is described is the File Description. FD in margin A identifies that what follows is a description of one of the files identified by the SELECT clause in the ENVIRONMENT DIVISION. After FD, place the name of the file to be described in margin B, followed by one or more of the clauses shown above, as applicable. The name of the file must be identical to that of the SELECT clause.

The only required clause is the LABEL clause. All others are optional. It is recommended that the other clauses are used to increase the efficiency of the compiler and for documentation.

All disk files have standard labels. All unit record files have omitted labels.

Army Standard: All possible entries are given for all file descriptions. All tape files must have standard labels unless created on another installation and used as input.

Under OS with no device model number specified in the SELECT statement, the label records are always standard regardless of the device to which the file is ultimately assigned. When used for magnetic tape, labels may appear at the beginning and the end of the file and are used to prevent the wrong program from accessing the information on the tape. When used for disk, labels are used to locate the beginning and end of sequential files. Since the system must know where a sequential file begins and ends on disk, labels must be used on disk; however, they are not mandatory for tape files. It is good programing practice to always use labeled files. This provides another check for the validity of the input or output data.

EXAMPLE (DOS & OS):

```
FILE SECTION.
FD  MASTER-IN
    LABEL RECORD IS STANDARD.
```

In this case, MASTER-IN was the name assigned to a file in a SELECT statement in the ENVIRONMENT DIVISION. This file was assigned to either a magnetic tape file or a disk file, if under DOS. All files have standard labels under OS. The LABEL RECORD IS STANDARD, which means to the system, generate standard labels according to my JCL instructions for output files or use the existing standard labels for input files.

If the file called MASTER-IN was assigned to a card reader, the FD would appear as:

EXAMPLE (DOS):

```
FILE SECTION.
FD  MASTER-IN
    LABEL RECORD IS OMITTED.
```

EXAMPLE (OS):

```
FILE SECTION.
FD  MASTER-IN
    LABEL RECORD IS STANDARD.
```

Since LABEL RECORDS can be used only for magnetic tape and disk under DOS, they must be OMITTED for card readers, punches, and printers. They may also be OMITTED under DOS for magnetic tape if a non-labeled tape is being used. OS always uses standard labels with device independence.

EXAMPLE (DOS):

```
FILE SECTION.
FD  PRINTOUT
    LABEL RECORD IS OMITTED.
```

EXAMPLE (OS):

```
FILE SECTION.
FD  PRINTOUT
    LABEL RECORD IS STANDARD.
```

DOS: In this case, a file called PRINTOUT, assigned to the 1403 printer in the ENVIRONMENT DIVISION, is described in its file description so that LABEL RECORDS ARE OMITTED.

OS: All devices have standard labels in OS regardless of the class.

There are other parts of the FD which may be used to reduce compile time for your program, and for more sophisticated programs, to reduce execution time. (See Appendix C, ANSI COBOL MANUAL.)

The BLOCK CONTAINS clause specifies the blocking factor--that is, how many records are to be read or written at once for each physical input/output operation. The BLOCK CONTAINS must be equal to 1 or omitted for unit-record devices (card readers, card punches, and printers), because a card is physically read, punched or a line is written each time the COBOL program READs or WRITEs a logical record. Physical READs and WRITEs take a lot of time when compared to the internal speed of the computer. So the programer, when using utility devices, can lump together several logical records into one physical record before it is read or written. This cuts down the number of READs and WRITEs on a device and speeds up the execution of the program. The BLOCK CONTAINS clause is needed in this case to tell the compiler that several logical records will be read or written each time the system performs a physical READ or WRITE to or from a device. As far as the programer is concerned, nothing changes in the program except the addition of the BLOCK CONTAINS clause indicating how many of the logical records are to be included in the block. On unit record devices, the logical record must be the same size as the physical record. If BLOCK CONTAINS is omitted, the system defaults to a blocking factor of 1.

EXAMPLE:

```
FILE SECTION.
FD  TAPE-FILE
    BLOCK CONTAINS 3 RECORDS
    LABEL RECORD IS STANDARD.
```

This entry would not write to the file called TAPE-FILE until 3 records had been built up into a block during processing via WRITE statements. If the file is input, then a block of 3 logical records is read and one is given to the program each time a READ statement for the file is executed.

The RECORDS CONTAINS clause specifies the length of each logical record in the file. A card, for example, contains 80 characters. Thus, the programer may write:

```
FILE SECTION.
FD  TAPE-FILE
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORDS OMITTED.
```

Note that, in the BLOCK CONTAINS clause, the programer may specify the number of characters instead of the number of records. If, on magnetic tape, each record was 40 characters in length, the FD could be written.

```
FILE SECTION.
FD  TAPE-FILE,
    BLOCK CONTAINS 120 CHARACTERS,
    LABEL RECORDS OMITTED.
```

or

```
FILE SECTION.
FD  TAPE-FILE,
    BLOCK CONTAINS 3 RECORDS,
    LABEL RECORD IS OMITTED.
```

or

```
FILE SECTION.
FD  TAPE-FILE,
    BLOCK CONTAINS 120 CHARACTERS,
    RECORD CONTAINS 40 CHARACTERS,
    LABEL RECORD IS OMITTED.
```

NOTE: Commas have no effect on the meaning of anything written. The programer may use either or both RECORD CONTAINS or BLOCK CONTAINS. Label records may be either STANDARD or OMITTED for tape files, depending on the desires of the programer. The reason that the RECORD CONTAINS clause may be omitted is that, following the FD clause, COBOL requires that the record for the file be described in detail. The compiler can, therefore, obtain this information from the description of the record (discussed after the FD clause in this text).

The DATA RECORD clause is used to give a name to the record for a file. It is the record which must be placed in memory; therefore, the description of the record is used to allocate storage. To initially specify what the record's name shall be, the programer uses the DATA RECORD clause.

Army Standard: To avoid duplicate data names, the data names chosen for records descriptions should be prefixed by a four character identifier record and file for that file. It should be in the form AANN where AA is an identifier unique for the file and NN is a two digit number unique for that record in the file. The first character of the file identifier should specify whether the file is input, output, or work ("I", "O", or "W"). The second character should describe the type of file (e.g., "M" for master file, "T" for transactions file, etc.).

EXAMPLE-1:

```
FILE SECTION.
FD  INPUT-FILE
    LABEL RECORD IS OMITTED
    DATA RECORD IS IT01-INPUT-RCD.
```

EXAMPLE-2:

```
FILE SECTION.
FD  MASTER-INVENTORY-FILE
    LABEL RECORD IS STANDARD
    DATA RECORD IS OM01-MASTER-INVENTORY-RCD.
```

In the first example, the name of the record for INPUT-FILE is IT01-INPUT-RCD. The prefix means that it is the first ("01") record description of an input ("I") transaction ("T") file. In the second example, the name of the record for MASTER-INVENTORY-FILE is OM01-MASTER-INVENTORY-RCD. The prefix means that it is the first ("01") record description of an output ("O") master ("M") file.

In the entire FD, there is only one period--the one at the end of all entries.

RECORD DESCRIPTIONS

A record is described by placing '01' in margin A, followed by the name of the record in margin B, followed by a description (a PICTURE) of the record.

EXAMPLE:

```
FILE SECTION.
FD  MASTER-FILE
    LABEL RECORD IS STANDARD
    DATA RECORD IS IM01-MASTER-RCD.
01  IM01-MASTER-RCD                PICTURE IS X(100).
```

In the above example, IM01-MASTER-RCD is the name of the record for MASTER-FILE. The prefix means that it is the first ("01") record description of an input ("I") master ("M") file. Since the label record is STANDARD, this file must either be on a disk or magnetic tape file. The line which begins with 01 (which is called a level number is used to allocate storage (memory) for the record called IM01-MASTER-RCD. In this case, IM01-MASTER-RCD has a length of 100 (there are 100 characters in the record).

PICTURE CLAUSE.

The PICTURE clause is used to allocate memory. There are 4 types of information which a PICTURE clause describes:

an alphanumeric field, designated by X's.
an alphabetic field, designated by A's.
a numeric field, designated by 9's and optionally a P, S, and/or V.
an edited field, designated by a combination of the characters
Z . , * $ - + S V B CR and DB.

Army Standard: (1) The PICTURE clause must be punched beginning in column 48.
(2) PICTURE is preferred abbreviated as PIC.
(3) When more than one clause is used to describe a data item, each clause must begin on a new card beginning in column 40.

An alphanumeric field (X) can contain any legal EBCDIC character. An <u>alphanumeric</u> literal (also called nonnumeric literal) is a character or series of characters enclosed in quotes and is handled as if it were an X (alphanumeric) field. For the continuation of an alphanumeric literal, use all columns through column 72 on the current card, release the card, and punch a dash (-) in column 7 of the next card. The alphanumeric literal is then continued by placing a quote somewhere in margin B and continuing the literal after the quote as though no continuation had been done. It is finished as a normal alphanumeric literal with a quote and period following the end of the alphanumeric literal.

A numeric field (9) is the only field which can be used in mathematical computations or tests and cannot contain any non-numeric characters.

An edit field is designed to edit a numeric or alphanumeric field (e.g., put in dollar signs, decimal points, commas, blanks, etc.)

In the example:

```
01  OM01-MASTER-RCD               PIC X(100).
```

OM01-MASTER-RCD is described as an alphanumeric field which can hold 100 characters. These characters can be any legal EBCDIC character, including any letter of the alphabet, zero thru nine, blanks, and any special characters such as commas, periods, asterisks, etc.

In order to refer to specific parts of a whole record, the record is "broken down" into smaller parts.

EXAMPLE: A punched card contains the name of an individual in columns 1-20;
the individual's age in columns 21-23;
spaces in columns 24-80.

To describe this card, the programer breaks the record into parts, using a higher level number to indicate that NAME, for example, is part of CARD-REC. The same holds true for AGE and FILLER. NOTE: Level numbers indicate the hierarchy of data within a record. The range is 01 through 49. Level 01 is highest level and indicates a record in the FILE SECTION. The higher the value of the number, the lower the level indicated. A field of a record may be assigned level number 3. Its subfields may be assigned level number 4.

Level numbers need not be consecutive, but must be sequential. The record level, must always be 01.

Army Standard: Subordinate data items of a record description all will be prefixed by the record prefix. This identifies them uniquely throughout the program. A fifth character may be added between the second and third characters if two files may have the same prefix, e.g., two input master files may be prefixed by IM101 and IM201 respectively.

```
FILE SECTION.
FD  CARD-FILE
    LABEL RECORD IS OMITTED
   DATA RECORD IS IT01-CARD-RCD.
01  IT01-CARD-RCD.
    05  IT01-NAME                 PIC X(20).
    05  IT01-AGE                  PIC 999.
    05  FILLER                        PIC X(57).
```

In the above example, IT01-NAME, IT01-AGE, and FILLER are all part of IT01-CARD-RCD, because the level number 05 is lower than 01. All numbers except 01, (which is discussed later) appear in margin B. Note that the number in parenthesis (called a duplication factor) indicates the number of times the preceding character is to be repeated. The above example could be written without the duplication factor:

```
FILE SECTION.
FD  CARD-FILE
    LABEL RECORD IS OMITTED
    DATA RECORD IS IT01-CARD-RCD.
01  IT01-CARD-RCD.
    05  IT01-NAME                 PIC XXXXXXXXXXXXXXXX.
    05  IT01-AGE                  PIC 999.
    05  FILLER                    PIC XXXXXXXXXXXXXXXXXXXXXXXXXX.
    05  FILLER                    PIC XXXXXXXXXXXXXXXXXXXXXX.
```

Notice that the FILLER with a length of 57 characters had to be broken into two pieces. That is, the number of characters used to represent the PICTURE cannot be longer than 30 characters, not the number of characters that PICTURE actually describes.

If a level is to be broken down into smaller parts, it contains no PICTURE clause and is referred to as a group item. The length of a group item is the sum of its subordinate items. To break IT01-NAME down into a first name field and a last name field, the programer could write:

```
01  IT01-CARD-RCD.
    05  IT01-NAME.
        10  IT01-FIRST-NAME       PIC X(10).
        10  IT01-LAST-NAME        PIC X(10).
    05  IT01-AGE                  PIC 9(3).
    05  FILLER                    PIC X(57).
```

IT01-NAME is now a group item and is still equal to a length of 20 characters because its parts' lengths add up to 20. In like manner, IT01-CARD-RCD is equal in length of 80 because its parts' lengths add up to 80.

The DATA RECORD clause need not be specified because the 01 entry is the record name and is implied just by being under its respective FD. It is good programing practice to include it.

Army Standard: As level numbers increase in record descriptions, they will go 01, 05, 10, etc. Columns for each level to begin in are 8, 12, 14, 16, etc., out to column 24. In no case will any level number be punched to the right of column 24.

WORKING-STORAGE SECTION.

The purpose of the FILE SECTION is to describe the files to be used in the program and to allocate memory required for the records of each of the files. The WORKING-STORAGE SECTION is used to allocate all memory needed for holding any data other than records described in the FILE SECTION. All storage in the FILE SECTION is transient. When a record is read, the information that was contained in the record area of the memory before the READ statement is gone forever. When a record is written, the information that was put in the record area of the memory is gone. WORKING-STORAGE, on the other hand, is permanent storage. Variables in this area only change when the program physically moves a new value into them. WORKING-STORAGE should be used for constants; e.g., headers, footers, holding areas, counters, and anything that needs to be kept around during the execution of the program. WORKING-STORAGE is divided into two areas: the 77's area and the 01's area. Each of these will be described in detail later. Because these areas are permanent storage, the compiler will allow you to initialize them to a given value. The record areas in the FILE SECTION cannot be initialized by the compiler since the areas change each time an input or output operation takes place.

Army Standard: All data names in the WORKING-STORAGE SECTION are prefixed by the characters WS-.

EXAMPLE:

```
WORKING-STORAGE SECTION.
77  WS-LINE-COUNT                    PIC S9(4)
                           USAGE IS COMP
                           VALUE IS ZEROS
                           SYNCHRONIZED.

01  WS-HEADING-LINE.
    05  FILLER                               PIC X(64)
                           VALUE IS SPACES.
    05  FILLER                               PIC X(69)
                           VALUE IS 'TITLE'.
```

A 77 data item is normally used for counters and hold areas. All 77 items appear immediately after WORKING-STORAGE SECTION and before the first 01 record entry. They are always elementary items and therefore cannot be subdivided with larger numbers like 78. In the example above, a 77 item named WS-LINE-COUNT is defined and initialized to a value of zero. The other area in WORKING-STORAGE is the 01 area. This area contains a series of 01 group or elementary items. The 01 group items may be broken down in a manner similar to the record descriptions in the FILE SECTION. In the example above, a group item named WS-HEADING-LINE is defined then broken down into two FILLERs. The first FILLER simply takes up 64 characters of storage and initializes them to spaces. The second FILLER takes up 69 characters of storage and initializes the first 5 to the characters TITLE. The other 64 characters of the 69 are initialized to spaces. Use of the value clause will be described in detail later. All items written as 77 entries can also be written as 01's. 01's can be elementary or group items.

EXAMPLE:

```
77  WS-LINE-COUNT                          PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
77  WS-PAGE-COUNT                          PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
77  WS-PERSON-COUNT                        PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
```

This could be coded as:

```
01  WS-COUNTERS.
    05  WS-LINE-COUNT                      PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
    05  WS-PAGE-COUNT                      PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
    05  WS-PERSON-COUNT                    PIC S9(4)
                               USAGE IS COMP
                               VALUE IS ZEROS
                               SYNC.
```

or also as:

```
0  WS-LINE-COUNT                        PIC S9(4)
                              USAGE IS COMP
                              VALUE IS ZEROS
                              SYNC.
01  WS-PAGE-COUNT                       PIC S9(4)
                              USAGE IS COMP
                              VALUE IS ZEROS
                              SYNC.
01  WS-PERSON-COUNT                     PIC S9(4)
                              USAGE IS COMP
                              VALUE IS ZEROS
                              SYNC.
```

Army Standard: The clauses included with a data item description should be in the following order:

REDEFINES
OCCURS
PICTURE (PIC)
USAGE
VALUE
JUSTIFIED
SYNCHRONIZED (SYNC).

All clauses mentioned above will be described later in detail.

All three methods yield the same result as far as defining the data items and making them available for use in the PROCEDURE DIVISION. The difference lies in the resultant storage allocation. It is enough for the beginning programer to know that for this type of data item the 77 is the most efficient and preferable of the three.

If the programer needs a numeric field and if that field will only be used for mathematical computation (not part of an input or output record, i.e., linecounter), the programer may specify that the USAGE of the numeric field is COMPUTATIONAL. This may be abbreviated as COMP. COMP fields are much more efficient for integer arithmetic than simple PIC 9 fields. You may not put in a V in the PICTURE. To print these fields is very time consuming, therefore, these should only be used for internal program counters. If the programer chooses to specify COMPUTATIONAL (for efficiency purposes), he should guarantee that this field begin on a correct boundry (A COMPUTATIONAL field, on IBM SYSTEM/360, is a numeric field which should begin either on a half or full word boundry). To direct that this field is to start on a correct boundry, the programer specifies SYNCHRONIZED (which may be abbreviated SYNC). See the above examples for this.

Army Standard: SYNCHRONIZED and COMPUTATIONAL are preferred to be abbreviated.

NOTE: A numeric field which is COMPUTATIONAL <u>must</u> include S (sign) as the leftmost character in the picture clause.

If SYNCHRONIZED is used to describe a field which does not need to be aligned on a boundry (e.g., PICTURE X), it is ignored.

In the following examples, WS-AGE, not being a COMPUTATIONAL field, does not have to be aligned on any boundry; thus SYNCHRONIZED is ignored.

EXAMPLE:

```
01  WS-CARD-RCD.
    05  WS-NAME.
        10  WS-LAST-NAME              PIC X(10).
        10  WS-FIRST-NAME             PIC X(10).
    05  WS-AGE                        PIC 999
                            SYNC.
```

If WS-AGE were not to be used as either a mathematical computation in the program or a numeric edited field on output, it could be defined as an alphanumeric (X) field:

```
    05   WS-AGE                       PIC X(3).
```

Note that each level entry ends with a period.

Any elementary item in WORKING-STORAGE (item with a PICTURE) may be assigned an initial value via the VALUE IS clause. Group items cannot be assigned a value and any data item under a REDEFINES clause cannot have a VALUE. REDEFINES is discussed in detail later. The only thing to keep in mind when assigning a VALUE to something is to remember how that thing is described in its PICTURE. Numeric items cannot be assigned an alphanumeric VALUE. Likewise, alphanumeric items should not be assigned a numeric literal VALUE; however, this is more acceptable than the first case. For example, one would not want to describe a counter as PIC 999 then try to initialize that counter to spaces. Spaces are not numbers and will not compute. COBOL has made things a little easier for programers to initialize variables. There exist a number of reserved words that represent specific values when associated with a data item. ZEROS is used with numeric PICTUREs. SPACES is used with alphanumeric PICTUREs.

EXAMPLES (Not in the order that they would be in a program):

```
01  WS-BLANK-LINE                    PIC X(133)
                        VALUE IS SPACES.
01  WS-COUNTER                       PIC 9(3)
                        VALUE IS ZEROS.
77  WS-LINE-COUNT                    PIC S9(4)
                        VALUE IS ZEROS.
01  WS-TITLE-1                       PIC X(5)
                        VALUE IS 'TITLE'.
77  WS-PAGE-COUNT                    PIC 999
                        VALUE IS 0.
```

The first example defines a 133 character block of memory and initializes it to spaces. The second example defines a 3 character numeric data item and initializes it to zeros. The third example defines a signed numeric variable of 4 characters and initializes it to zeros. The S in front of the 9 indicates that this variable may be negative or positive. The absence of an S indicates that this variable will always be positive and any value that is moved to that item is made positive. The fourth example defines a 5 character alphanumeric item and is initialized to a value of 'TITLE'. This example uses another form of initialization: the literal. Literals can be either alphanumeric (enclosed in quotes) for X PICTUREs or numeric for 9's PICTUREs. PICTUREs are described in detail in the next section. If a numeric PICTURE has an S and a numeric literal is used, then that numeric literal must have a + or - in front of it.

EXAMPLE:

```
77  LINE-COUNT                        PIC S9(4)
                            USAGE IS COMP
                                                    VALUE IS +100
                            SYNC.
```

It is important to remember that a literal, whether numeric or alphanumeric, cannot exceed the length of the associated PICTURE. It was mentioned before that a numeric value should not be assigned to an alphanumeric PICTURE. ZEROS cannot be associated with an X PICTURE. However, the following is legal and desirable sometimes:

EXAMPLE:

```
77  NUMERIC-LITERAL                   PIC X(10)
                            VALUE IS '0123456789'.
```

This example defines a 10 character alphanumeric data item and initializes it to the character string 0123456789. Anything placed within the quotes of the VALUE clause in an alphanumeric literal is legal and allowable. They are just another string of characters as far as the compiler is concerned. It should be remembered that one still cannot do arithmetic with the above example because the PICTURE is alphanumeric. If the literal used with an alphanumeric item is shorter than the PICTURE, the literal is placed in the left of the field and padded to the right with spaces.

RESTRICTION OF THE VALUE CLAUSE: IT CAN ONLY BE USED IN THE WORKING-STORAGE SECTION; IT CANNOT BE USED IN THE FILE SECTION.

Numeric literals may contain numeric digits (0 thru 9), a sign if the PICTURE clause has an S and a decimal point to indicate proper decimal point alignment.

EXAMPLES:

```
77  WS-PAGE-COUNTER                         PIC 999
                                  VALUE IS 1.
77  WS-TAX-RATE                             PIC SV99
                                  VALUE IS +.18.
77  WS-TOTAL-TAX                            PIC S9(4)V99
                                  VALUE IS +0.
```

Note that even when the literal has a decimal point in it, a period still ends the data item description giving two periods in the description.

If the literal used with a numeric PICTURE is shorter than the PICTURE, then the literal is placed in the right of the field and padded to the left with zeros. The only exception to this is when a V is included in a numeric PICTURE.

EXAMPLE:

```
77  WS-TAX-RATE                             PIC SV99
                                  VALUE IS +.23.
77  WS-NET-PAY                              PIC S9(5)V99
                                  VALUE IS +25.
```

In the first example, TAX-RATE is defined as a two character numeric variable with the implied decimal point to the left of the two digits. There is no actual decimal point in the field, but arithmetic is done as though it existed. Any number moved to this field will be adjusted so the implied decimal point is in the appropriate place. For example, if one moved the integer 33 to WS-TAX-RATE, WS-TAX-RATE would have zeros in it because 33 implied 33.00 and WS-TAX-RATE only has places for the numbers to the right of the decimal point. However, if one moved 33.25 to WS-TAX-RATE, WS-TAX-RATE would contain 25 with the implied decimal point to the left of the numbers. Decimal points are not numbers so they cannot be physically in the field that has the number being used in the computation. The V is used only for adjusting the significance of the digits. If it is possible for a move or computation to result in losing high order digits, a message will be printed in the error listing warning the programer of the possibility. In the second example, WS-NET-PAY is defined as having seven characters of memory and is initialized to a value of 0002500 with the right two digits implied to be decimals. Notice that after the adjustment was made for the decimal point, the field was padded in both directions with zeros.

EDITED FIELDS

An EDIT field is designed to put symbols such as $, . * + - in the correct place to make a numeric field more understandable. The following symbols may appear in a report field:

List 1: S V 9
List 2: Z , $. + - * B CR DB

If one or more symbols from List 2 are used in a PICTURE clause, the field becomes a numeric EDIT field and cannot be used in any mathematical computation. If only the symbols in List 1 are used, the field remains a numeric field and may be used in any mathematical computation. The only way to get information into a EDIT field is to MOVE a numeric field TO an EDIT field (this instruction is given in the PROCEDURE DIVISION). Usually EDIT fields are used as part of an output description for a print file.

CHARACTER	MEANING
Z	Zero supress non-sigificant leading zeros.
,	Put a comma here if there are any significant digits to the left.
$	If one occurrence, put a dollar sign here; if multiple occurrences, put a dollar sign to the left of the first significant digit.
.	Decimal point.
+	Print a plus sign if the number moved into the field is positive; print a minus sign if the field is negative.
-	Print a space if the number moved into the field is positive; print a minus sign if the field is negative.
*	Check protection: replace non-significant leading zeros with asterisks.
V	Implied decimal position. This does not take up a print position or a column on input.
S	Keep track of the sign for numeric fields (do not use S if any character from list 2 appears in the PICTURE clause).
B	Generate a blank wherever it is inserted.
CR	Prints two spaces if the value of the number moved into the field is positive. Prints as CR if the value is negative.
DB	Prints two spaces if the value of the number moved into the field is positive. Prints as DB if the value is negative.

EXAMPLES OF EDITED MOVES:

V's are shown for explanation only

PICTURE	VALUE OF DATA	EDITED RESULT
$$$$.99	V12	$.12
$$,$$9.99	500V25	$500.25
$$,$$9.99	1500V25	$1,500.25
$Z,ZZ9.99	1500V25	$1,500.25
$Z,ZZ9.99	-V25	$ 0.25
$Z,ZZ9.99-	-V25	$ 0.25-
$Z,ZZ9.99-	1V50	$ 1.50
$*,**9.99-	25V75	$***25.75
$*,**9.99+	25V75	$***25.75+
$*,**9.99+	-25V75	$***25.75-
$*,**9.99BCR	25V75	$***25.75
$*,**9.99BCR	-25V75	$***25.75 CR
$*,**9.99CR	-25V75	$***25.75CR
$*,**9.99BDB	25V75	$***25.75
$*,**9.99BDB	-25V75	$***25.75 DB
$*,**9.99DB	-25V75	$***25.75DB
$+,+++.99	25V75	$ +25.75
$+,+++.99	-25V75	$ -25.75
$-,---.99	25V75	$ 25.75
$-,---.99	-25V75	$ -25.75

ARRAYS (TABLES)

An array is nothing more than a series of identical data items. Instead of having to define all of them individually, COBOL allows one to say that one thing OCCURS n times. All that the programer has to do is define what one of the things looks like. The things are then referenced by use of an index or a subscript. The index or subscript simply tells the compiler which of the things you want to look at.

EXAMPLE:

```
01  WS-TABLE.
    05  WS-TABLE-ENTRY                  OCCURS 25 TIMES
                              INDEXED BY TABLE-INDEX.
        10  WS-NAME                     PIC X(20).
        10  WS-SOC-SEC-NUM              PIC X(9).
```

This example defines an array named WS-TABLE made up of 25 WS-TABLE-ENTRY's. Each WS-TABLE-ENTRY is broken down into a WS-NAME and WS-SOC-SEC-NUM. The entire array is 580 characters long. ((20+9) x 25) This array is indexed by WS-TABLE-INDEX. How to reference arrays in the PROCEDURE DIVISION is covered in the section on the PROCEDURE DIVISION.

RESTRICTIONS ON ARRAYS: The OCCURS clause cannot be used with 77 or 01 items. Any item containing an OCCURS clause or subordinate to an item containing an OCCURS clause cannot have a VALUE clause.

REDEFINES CLAUSE

The same area of memory may be described in more than one way by using the REDEFINES clause. Many times it is necessary to use a particular data item in several ways (alphanumeric or numeric). This is done with the REDEFINES clause.

EXAMPLE:

```
01  WS-DISK-RCD.
    05  WS-PAY                          PIC S9(5)V99
                                  VALUE ZEROS.
                                        REDEFINES WS-PAY
    05  WS-PAY-1
                                  PIC X(7).
01  WS-CARD-RCD.
    05  WS-NAME-1.
        10  WS-LAST-NAME-1              PIC X(12).
        10  WS-FIRST-NAME-1             PIC X(8).
    05  WS-NAME-2                       REDEFINES WS-NAME-1.
        10  WS-FIRST-NAME-2             PIC X(8).
        10  WS-LAST-NAME-2              PIC X(12).
```

In the first example, WS-PAY is simply REDEFINED as alphanumeric by WS-PAY-1. This type of redefinition is particularly useful when checking the validity of numeric fields. More of this will be covered in the discussion of the PROCEDURE DIVISION. In the second example, the storage assigned to WS-NAME-1 is defined in two ways: WS-LAST-NAME then WS-FIRST-NAME and WS-FIRST-NAME then WS-LAST-NAME. When the data coming into the program can be of several different forms, the REDEFINES clause is very helpful.

RESTRICTIONS ON THE REDEFINES CLAUSE:

1. Redefinition within the elements subordinate to an OCCURS clause will generate a warning.

2. Be sure to make the two items (the object and subject of the REDEFINES) the same length.

3. REDEFINES cannot be specified in an 01 entry in the FILE SECTION.

4. If A REDEFINES B, then both A and B must have the same level number and there must be no intervening level numbers having a level number equal to or less than A and B's level numbers. For example, if A and B were both 05

entries, then no entry with 05 or less could be between A and B. In example 2 above, there are 10's between NAME-1 and NAME-2. Both NAME-1 and NAME-2 are 05 entries.

88 LEVEL ITEMS

An 88 level item may be defined under any data item, both elementary and group level. It has no PICTURE clause. Instead it is used to describe possible values or ranges of values and to associate a name with the description.

EXAMPLE:

```
05  IT01-TRANSACTION-CODE      PIC X.
    88 ADDITION                VALUE 'A'.
    88 CHANGE                  VALUE 'C'.
    88 DELETION                VALUE 'D'.
```

In this example, the elementary data item IT01-TRANSACTION-CODE is described as having the possible value of A, C, or D. If one of these values occurs, the name associated with them is true. See the discussion of the IF statement to see how to use the 88 in the PROCEDURE DIVISION. If any other value other than the ones described in the 88's occurs, none of the 88's are true, but no error condition is raised.

EXAMPLE:

```
01  WS-PAY                     PIC 9(6)V99.
    88 SMALL                   VALUE .00 THRU 100.00.
    88 MEDIUM                  VALUE 100.01 THRU 300.00.
    88 LARGE                   VALUE 300.01 THRU 1000.00.
    88 VERY-LARGE              VALUE 1000.01 THRU 999999.99
```

Note in the first example that the values associated with the IT01-TRANSACTION-CODE are enclosed in quotes because the PICTURE of IT01-TRANSACTION-CODE is alphanumeric. In the second example, the VALUEs are numeric literals because the PICTURE of WS-PAY is numeric. 88 level entries can be in both the FILE SECTION and the WORKING-STORAGE SECTION. They should not be used under 77 level items. Ranges may be used with alphanumeric items, but care should be exercised.

EXAMPLE:

```
05  WS-DIVISION-CODE           PIC XX.
    88 DIVISION-1              VALUE '01' THRU '02'.
    88 DIVISION-2              VALUE '03' THRU '04'.
```

What is being described is the internal machine representation of '01' thru the internal representation of '02' and all intervening characters. This type of 88 is very machine dependent and can be the cause of obscure errors in processing.

THE PROCEDURE DIVISION

The PROCEDURE DIVISION contains the instructions that manipulate the data described in the DATA DIVISION. The PROCEDURE DIVISION is made up of a series of paragraphs. Each paragraph begins with a name (the paragraph name) followed by a period followed by the body of the paragraph. The body is made up of one or more COBOL PROCEDURE DIVISION sentences. A COBOL sentence is one or more COBOL verbs and their associated parts. A sentence is ended by a period. All paragraph names begin in margin A. All sentences begin in margin B.

Army Standard: All paragraph names will be preceded by a four digit number. Paragraphs will be in ascending order by this number. The numbers will be incremented by 10 to allow insertion of new paragraphs in the future.

EXAMPLE:

```
PROCEDURE DIVISION.
0010-PARA-NAME.
    OPEN INPUT  CARD-READER
        OUPUT PRINTER.
    PERFORM 0020-READ-AND-PRINT THRU 0020-RAP-EXIT
        UNTIL WS-EOF-SWITCH = 'OFF'.
    CLOSE CARD-READER,
        PRINTER.
    STOP RUN.
0020-READ-AND-PRINT.
    READ CARD-READER
        AT END MOVE 'OFF' TO WS-EOF-SWITCH
        GO TO 0020-RAP-EXIT.
    MOVE CARD-RECORD TO PRINT-RECORD.
    WRITE PRINT-RECORD AFTER ADVANCING 2 LINES.
0020-RAP-EXIT.  EXIT.
```

In the above example, there are three paragraphs. The first starts with 0010-PARA-NAME and ends with STOP RUN. The second paragraph begins with 0020-READ-AND-PRINT and ends with WRITE PRINT-RECORD AFTER ADVANCING 2 LINES. The third begins with 0020-RAP-EXIT and ends with EXIT (this has only one sentence in it). EXIT will be explained later, but for now, the EXIT paragraph is considered to be the end of the preceding paragraph. What this program actually does is immaterial at this point. Each of the verbs (OPEN, PERFORM, CLOSE, STOP, READ, MOVE, GO, WRITE) will be discussed in detail later. When the PROCEDURE DIVISION is executed, the first statement after PROCEDURE DIVISION is executed first. Because of this, no paragraph name is needed immediately after PROCEDURE DIVISION. ANSI requires para name. Normally paragraphs represent blocks of code that do one job. In the example, everything from 0020-READ-AND-PRINT thru 0020-RAP-EXIT is involved in the reading and writing of data records. Everything from the PROCEDURE DIVISION thru the STOP RUN is involved in preparing the files to be read and written, controlling the reading and writing, then cleaning up after it has all been done. Each paragraph has its own job and doesn't depend on anything else to do that job for it.

The PROCEDURE DIVISION's verbs have been divided into three classes:

1. Process verbs - words which initiate manipulation of data in the program.

2. I/O verbs - words which initiate data transfer from the outside world into the program or vice versa.

3. Logic control verbs - words which affect the order of statement execution.

Each of these categories of verbs are discussed below.

PROCESS VERBS

MOVE

The form of the MOVE verb is:

```
MOVE {data-name-1} TO  data-name-2 [data-name-3  ...]
     {literal    }
```

This verb copies information from one data-name or literal to one or more data-names. There are several restrictions on moving data from one area to another that are connected to the PICTUREs of the respective data-items. See Table 1. An example of the MOVE statement is:

EXAMPLE:

```
MOVE IM01-NAME TO WS-NAME.

MOVE IM01-DIVISION TO WS-DIVISION
                      WS-DIVISION HOLD.
```

In the first example, the data-item IM01-NAME is copied to WS-NAME. When the move is complete, both IM01-NAME and WS-NAME contain the information that IM01-NAME had contained before the MOVE was executed. In the second example, the information in IM01-DIVISION is copied to both WS-DIVISION and WS-DIVISION-HOLD. When the MOVE is complete, all three data items have the same information in them.

Another form of the MOVE statement is:

```
MOVE {CORRESPONDING} group-item-1 TO group-item-2.
     {CORR         }
```

This form of the MOVE verb copies all of the data-names under group-item-1 that have the same data-names under group-item-2. When the MOVE CORRESPONDING is complete, anything from all of the data-items under group-item-1 to none of the data-items under group-item-1 may have been moved.

Army Standard: Since non-unique names are not allowed and the move corresponding depends on non-unique names, the move corresponding statement is not allowed.

EXAMPLE:

```
FILE SECTION.
.
.
01  IM01-MASTER-RCD.
    05  IM01-NAME                        PIC X(20).
    05  IM01-SSN                         PIC X(9).
    05  IM01-RANK                        PIC X(3).
    05  FILLER                           PIC X(48).
WORKING-STORAGE SECTION.
.
.
01  WS-DETAIL-LINE-1.
    05  FILLER                           PIC X(48)
                               VALUE IS SPACES.
    05  WS-NAME                          PIC X(20).
    05  FILLER                           PIC X(3)
                               VALUE IS SPACES.
    05  WS-RANK                          PIC X(3).
    05  FILLER                           PIC X(3)
                               VALUE IS SPACES.
    05  WS-SSN                           PIC X(9).
    05  FILLER                           PIC X(47)
                               VALUE IS SPACES.
01  WS-DETAIL-LINE-2.
    05  FILLER                           PIC X(48)
                               VALUE IS SPACES.
    05  IM01-NAME                        PIC X(23).
    05  IM01-RANK                        PIC X(6).
    05  IM01-SSN                         PIC X(56).
.
.
PROCEDURE DIVISION.
.
.
MOVE CORRESPONDING IM01-MASTER-RCD TO WS-DETAIL-LINE-1.
MOVE CORRESPONDING IM01-MASTER-RCD TO WS-DETAIL-LINE-2.
```

In the first MOVE CORRESPONDING statement, nothing would have been moved and no error message generated. This is one weakness of the MOVE CORRESPONDING statement. None of the data-items are named identically under group-item-1 and group-item-2. In the second MOVE CORRESPONDING, all of the data-item in MASTER-RECORD are moved in DETAIL-LINE-2. Note that instead of defining all of the FILLERs between the data-items as in DETAIL-LINE-1, the data-items in DETAIL-LINE-2 are simply defined as containing those FILLERs. When the

IM01-NAME defined in IM01-MASTER-RECORD is moved to the IM01-NAME defined in WS-DETAIL-LINE-2, the information is moved into the left most of the 23 character receiving field and the remaining 3 characters are filled with blanks. Similar actions accompany the other two moves. This type of definition is less efficient than the first because the trailing characters in all of the receiving fields are filled with blanks each time the move statement is executed. In the first example, the blanks are put in only once during the compilation and remain there all during the execution of the program with no modification. It is normally a good idea to make receiving fields the same size as sending fields.

NOTE: If any of the data item names are defined with the same data name in two or more areas and need to be referenced individually, they may be referenced by qualifying the item with the qualifier OF.

EXAMPLE:

MOVE RANK OF MASTER-RECORD TO RANK-HOLD.

Army Standard: Qualification of data names will not be used.

TABLE 1

LEGAL MOVES:

1. (A) an alphabetic field may be moved to another alphabetic field. The movement is from the leftmost byte of the receiving field to the right. If the receiving field is larger, the remaining positions will be filled with spaces. If the receiving field is shorter than the sending field, truncation of the right most position(s) will occur.

2. (X) an alphanumeric field may be moved to another alphanumeric field. The same rules apply here as they do in an alphabetic field.

3. (9) a numeric field may be moved to another numeric field. However, the movement is accomplished by decimal point alignment. If the receiving field is smaller, truncation will occur and if the receiving field is larger, zeros will fill in the remaining core area. Exactly how the truncation takes place will be explained later.

4. A numeric or alphabetic field may be moved to an alphanumeric field. When numeric (PIC 9's) date fields are moved to alphanumeric (PIC X's) data fields, the move is treated as an alphanumeric move and follows the rules shown under rule 1.

5. An edited field may be moved to an alphanumeric field.

6. A numeric field may be moved to an edited field.

7. An alphabetic field may be moved to an alphanumeric edited field.

EXAMPLE:

MOVE A-FIELD TO B-FIELD.
If B-Field were alphanumeric, A-Field could be alphabetic, alphanumberic or edited.
If B-FIELD were numeric, A-FIELD could only be numeric.
If B-FIELD were alphabetic, A-FIELD could only be alphabetic.
If B-FIELD were edited, A-FIELD could only be numeric.

In the above examples, A-FIELD remains unchanged; B-FIELD becomes a "copy" of A-FIELD.

EXAMPLE:

MOVE SPACES TO OR01-PRINTER-RCD.

In this case, OR01-PRINTER-RCD must be defined as PICTURE X (alphanumeric) or PICTURE A (alphabetic), because spaces is recognized by the compiler as 1 or more "blanks", which is a character, not a number, and can be placed only into a PICTURE (A) or (X) fields.

NOTE: All group level items (items which are broken down into smaller parts) are handled as PICTURE X fields regardless of the PICTURE of the subordinate elements of the field are described.

EXAMPLE:

```
01  WS-OUT-RCD.
    05  WS-NUMBER              PIC 9(6).
    05  FILLER                 PIC X(127).
```

In this example, WS-OUT-RCD is alphanumeric (PICTURE (X)) because it is a group item. Its subordinate parts are WS-NUMBER and FILLER, both elementary items because they have PICTURE clauses. WS-OUT-RCD's length is the sum of its parts (133 characters). The programer can legally write MOVE SPACES TO WS-OUT-RCD because WS-OUT-RCD is a group item and therefore alphanumeric even though the first six positions are defined as numeric. If this move was done, then WS-NUMBER cannot be used in arithmetic computations because the information in it is not numeric and will cause a 'data exception' (trying to use non-numeric data as though it were numeric). Numeric information must be moved to WS-NUMBER before WS-NUMBER can be used as a number.

If the two items are alphanumeric and not the same size, one of two things will happen:

1. If the smaller item is moved to the larger, the data in the smaller item is put as far left as possible in the larger field. The remaining characters of the larger item will be filled with spaces.

2. If the larger item is moved to the smaller, the smaller item is filled up with the information from the larger starting at the left of both fields. The extra characters in the larger field are ignored (truncated).

EXAMPLE:

```
01  IM01-CARD.
    05  IM01-NAME             PIC X(20).
    05  IM01-SSN              PIC X(9).
    05  IM01-ADDRESS          PIC X(40).
    05  FILLER                PIC X(11).

01  WS-DETAIL-LINE.
    05  FILLER                PIC X(33).
    05  WS-NAME               PIC X(25).
    05  WS-SSN                PIC X(15).
    05  WS-ADDRESS            PIC X(30).
    05  FILLER                PIC X(32).
    .
    .
    .
MOVE IM01-NAME TO WS-NAME.
MOVE IM01-SSN TO WS-SSN.
MOVE IM01-ADDRESS TO WS-ADDRESS.
```

IM01-NAME is moved to the left most twenty characters of WS-NAME and the remaining five characters are filled with spaces. IM01-SSN is likewise put in the leftmost nine characters of WS-SSN and the remaining six characters are filled with spaces. Since IM01-ADDRESS is larger than WS-ADDRESS, only the leftmost thirty characters of IM01-ADDRESS are moved to WS-ADDRESS.

All numeric fields have a decimal point either implied (with a 'V'), assumed (if no other decimal point is given, it defaults to the right of the rightmost digit), or indented (as a '.'). Movement from one numeric field to another takes place after decimal point alignment. One or both of these cases may apply.

1. After decimal point alignment, the digits are moved from the sending field into the receiving field, beginning at the left of the decimal point and proceding left. If the sending field has more digits to the left of the decimal point than the receiving field, only as many as will fit are moved. The rest are truncated. If the sending field has fewer digits to the left of the decimal point than the receiving field, then the extra digits are filled with zeros.

2. If the sending field has more digits to the right of the decimal point than the receiving field, then only as many as will fit are moved. The rest are truncated. If the sending field has fewer digits to the right of the decimal point than the receiving field, then the extra digits are filled with zeros.

EXAMPLES:

```
77  WS-NUM-1                         PIC 99V9
                           VALUE 25.1.
77  WS-NUM-2                         PIC 999V99.
77  WS-NUM-3                         PIC V999.
77  WS-NUM-4                         PIC 99.

MOVE WS-NUM-1 to WS-NUM-2,
                 WS-NUM-3,
                 WS-NUM-4.
```

After the moves, the value of WS-NUM-1 is 25.1, WS-NUM-2 is 025.10, WS-NUM-3 is .100, and WS-NUM-4 is 25.

To the left of the decimal point, the data is right justified. Any excess high order positions are filled with 0's. To the right of the decimal point, they are left justified. Any excess low order positions are filled with 0's. Truncation occurs as follows: To the left of the decimal point, the high order is truncated. To the right of decimal point, the low order is truncated.

ADD

There are three forms of the ADD verb. The first form is normally used to add one or more things to a sum.

```
ADD {data-name-1        } [data-name-2        ...
    {numeric-literal-1  } [numeric-literal-2

     data-name-m      ]  TO  data-name-n [ROUNDED]
     numeric-literal-m]

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

EXAMPLE 1:

```
ADD WS-FICA-TAX WS-CITY-TAX WS-LOCAL-TAX WS-STATE-TAX TO WS-TOTAL-TAX.
```

EXAMPLE 2:

```
ADD 1 to WS-LINE-COUNT.
```

The first example adds four individual taxes to the total already in WS-TOTAL-TAX yielding a new total. This list of taxes could also be added to several other things in the same statement by simply listing the things after WS-TOTAL-TAX. Each of the resultant fields may have the ROUNDED clause included after it. If ROUNDED is used, the result is rounded according to the receiving field's PICTURE. After decimal point alignment, the number of places to the right of the decimal point of the result of an arithmetic computation is compared with the number of places allowed for the result in the receiving field. If more places are in the result than can be accommodated for in the receiving field, normally the extra places are simply discarded (truncated). If ROUNDED is specified, then the most significant digit of the excess is examined. If this digit is 5 or more, then the least significant digit of the part that is going to be used will be incremented by one. If this digit is 4 or less, then the number remains the same and the unneeded low order digits are discarded. If ON SIZE ERROR is specified after the list of receiving fields, then if the result of the computation is too large to fit in the specified PICTURE, the list of verbs after the ON SIZE ERROR is executed. Whether the ON SIZE ERROR clause is specified or not and a high order truncation condition occurs, the truncated result is still put in the receiving fields. In the second example, WS-LINE-COUNT is incremented by 1.

Army Standard: Use of ON SIZE ERROR and ROUNDED should be avoided when possible.

The second form of the ADD verb is used to add two or more things into one or more receiving fields. ROUNDED and ON SIZE ERROR may be specified for each of the receiving fields causing the same treatment as in the first form of the ADD verb.

```
ADD {data-name-1          data-name-2         [...]} GIVING data-name-m [ROUNDED]
    {numeric-literal-1    numeric-literal-2        }

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

EXAMPLE:

```
ADD IM01-NUMER-OF-DEPENDENTS 1 GIVING WS-FAMILY-SIZE,
    ON SIZE ERROR DISPLAY 'FAMILY SIZE TOO BIG'
        GO TO 0100-PARA-EXIT.
```

In this example, IM01-NUMBER-OF-DEPENDENTS and the numeric-literal 1 are added together and the result is placed into WS-FAMILY-SIZE. If WS-FAMILY-SIZE cannot contain the number of digits of the result of the computation, then an ON SIZE ERROR condition exists and the verbs after ON SIZE ERROR until the next period are executed. If ON SIZE ERROR had not been specified and the condition arose during the computation, the result would be truncated and the next statement executed. An advantage of the GIVING option is that none of the original values, except the result, are changed by the ADD.

The third form of the ADD verb adds all of the items under group-item-1 to all of the identically named items under group-item-2. ROUNDED and ON SIZE ERROR operate identically as stated before.

```
ADD CORRESPONDING group-item-1 TO group-item-2 [ROUNDED]
    CORR

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

EXAMPLE:

```
ADD CORRESPONDING IM01-MASTER-PAY-RECORD TO WS-MASTER-PAY-TOTALS.
```

This example will add all of the items under IM01-MASTER-PAY-RECORD to the identically named items under WS-MASTER-PAY-TOTALS truncating the results if necessary.

Army Standard: Add corresponding will not be used.

SUBTRACT

SUBTRACT has three forms. They are:

```
SUBTRACT {data-name-1        } [data-name-2         ...] FROM  data-name-m [ROUNDED]
         {numeric-literal-1  } [numeric-literal-2      ]

    [data-name-n [ROUNDED] ...]

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]

SUBTRACT {data-name-1         data-name-2         ...} FROM  data-name-m
         {numeric-literal-1   numeric-literal-2      }       numeric-literal-m

    GIVING data-name-n [ROUNDED]

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]

SUBTRACT {CORRESPONDING} group-item-1 FROM group-item-2 [ROUNDED]
         {CORR         }

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

In the first form of the SUBTRACT verb, one or more items are added up then the total is subtracted from the item or items following the FROM. ROUNDED and ON SIZE ERROR work identically to the ADD statement.

Army Standard: Use of ON SIZE ERROR and ROUNDED should be avoided where possible.

EXAMPLES:

```
SUBTRACT 1 FROM WS-TOTAL-PROPLE.
SUBTRACT WS-FICA-TAX WS-SUC-SEC-TAX FROM WS-GROSS-PAY ROUNDED
    ON SIZE ERROR MOVE '1' TO WS-ERROR-FLAG
        GO TO 0150-PARA-EXIT.
```

In the second form of the SUBTRACT verb, one or more items are added up then the total is subtracted from the item after the FROM clause and the final result is put into the one or more data-items following the GIVING clause ROUNDED and ON SIZE ERROR work identically to the ADD statement.

EXAMPLES:

```
SUBTRACT 1 FROM WS-FAMILY-SIZE GIVING WS-NUMBER-OF-DEPENDENTS.
SUBTRACT WS-FICA-TAX WS-SOC-SEC-TAX FROM WS-GROSS-PAY GIVING WS-NET-PAY.
```

In both the examples, the item following the FROM is not changed by the computation.

In the third form of the SUBTRACT verb, all of the items under group-item-1 are subtracted from the identically named items under group-item-2. ROUNDED and ON SIZE ERROR work identically to the ADD statement.

EXAMPLE:

```
SUBTRACT CORR IM03-INVENTORY-ADJUSTMENT-RECORD FROM
WS-TOTAL-INVENTORY-DISCREPANCY.
```

Army Standard: Subtract corresponding will not be used.

MULTIPLY

This verb is used to multiply one by another. It has two forms.

```
MULTIPLY {data-name-1      } BY data-name-2          [ROUNDED]
         {numeric-literal-1}

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]

MULTIPLY {data-name-1      } BY {data-name-2      } GIVING data-name-3 [ROUNDED]
         {numeric-literal-1}    {numeric-literal-2}

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

In the first form of the MULTIPLY verb, a number (either a numeric-literal or a data-name) is multiplied by another data-name and the result is put in the data-name after BY. ROUNDED and ON SIZE ERROR work in the same manner as in the ADD statement.

Army Standard: Use of ROUNDED and ON SIZE ERROR should be avoided where possible.

EXAMPLE:

```
MULTIPLY WS-TAX-RATE BY WS-GROSS-PAY.
```

This example multiplies the contents of WS-TAX-RATE by the contents of WS-GROSS-PAY and puts the result in WS-GROSS-PAY.

The second form of the MULTIPLY verb is used to multiply two values together putting the result into a data-name. ROUNDED and ON SIZE ERROR work in the same way as in the ADD statement.

EXAMPLE:

```
MULTIPLY WS-GROSS-PAY BY WS-TAX-RATE GIVING WS-FEDERAL-TAX ROUNDED
    ON SIZE ERROR DISPLAY 'FEDERAL TAX OVERFLOW'
        GO TO 0210-PARA-EXIT.
```

DIVIDE

The DIVIDE statement has two forms:

```
DIVIDE {data-name-1        } INTO data-name-2 [ROUNDED]
       {numeric-literal-1  }

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]

DIVIDE {data-name-1       } {INTO} {data-name-2       } GIVING data-name-3 [ROUNDED]
       {numeric-literal-1 } {BY  } {numeric-literal-2 }

    [REMAINDER data-name-4]

    [ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
    ELSE (period)]
```

The first form of the DIVIDE verb is used to divide a number or data-name into another data-name. ROUNDED and ON SIZE ERROR work the same way as in the ADD statement. The result is left in the second data-name.

Army Standard: Use of ROUNDED and ON SIZE ERROR should be avoided where possible.

EXAMPLES:

```
DIVIDE 10 INTO WS-PAGE-TOTAL.
DIVIDE WS-TAX INTO WS-PAY ROUNDED.
```

In both examples, the result is left in the second variable (after the INTO).

The second form of the DIVIDE verb divides a number or data-name into or by another number or data-name giving a third number or data-name. ROUNDED and ON SIZE ERROR work in the same manner as in the ADD statement.

EXAMPLES:

DIVIDE WS-TAX INTO WS-PAY GIVING WS-TAX-RATE.
DIVIDE A BY B GIVING C ROUNDED.

In both examples, the result is placed in the variable after the GIVING leaving the first two variables unchanged.

COMPUTE

COMPUTE identifier-1 [ROUNDED] = $\left\{\begin{array}{l}\text{arithmetic-expression}\\ \text{identifier-2}\\ \text{literal-1}\end{array}\right\}$

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

The COMPUTE statement gives COBOL powers similar to that of FORTRAN: a complete arithmetic expression may be evaluated and the result placed in identifier-1.

There are five (5) arithmetic operators that may be used in arithmetic expressions. Each is represented by a specific character or character combination that must be preceded by a space and followed by a space, except that a sign (- or +) must not be preceded by a space when it follows a left parenthesis.

ARITHMETIC OPERATOR	MEANING
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Parenthesis may be used in arithmetic expressions to specify the order in which elements are to be evaluated. Expressions within parenthesis are evaluated first. When parenthesis are not used or parenthetical expressions are at the same level of inclusiveness, the following heirarchical order is implied:

1. - or + (indicating negative or positive number)
2. **
3. * and /
4. + and -

When the order of consecutive operations on the same heirarchical level is not completely specified by parentheses, the order of operation is from left to right.

ON SIZE ERROR and ROUNDED operate identically to the ADD statement.

Army Standard: Use of the ON SIZE ERROR and ROUNDED should be avoided where possible.

EXAMPLE: COMPUTE WS-RESULTS ROUNDED = (A + B - C * D) - (-E / F + (G * H ** I))

ORDER OF OPERATION:

H is raised to the Ith power
G is multiplied by the result of (H**I)
E is negated
F is divided into E
The result of -E/F is added to G*H**I
C is multiplied by D
A is added to B
C*D is subtracted from A+B
The result of -E/F+(G*H**I) is subtracted from the result of A+B-C*D
This result is rounded up and the result if placed into WS-RESULTS.
A B C D E F G H and I remain unchanged.

NOTE: Army naming convention is dropped for this example for illustrative clarity.

INPUT-OUTPUT

READ

The form of the READ statement is:

READ file-name

AT END any series of COBOL verbs and associated parts except the IF or IF-ELSE ended by a period.

This verb reads any sequential file (cards, disk, or tape file) and places the information that was read into the record description under the file-name in the FD. Each time this verb is executed, one record is read from the file. After the last record has been READ, the next time the READ statement is executed, the AT END series of verbs is executed. Care should be exercised to insure that once the AT END condition arises the file is never READ again except when another file immediately follows the one just processed. If a record was actually READ, the period (after the AT END series of verbs) is executed.

EXAMPLE:

```
READ CARD-FILE
    AT END MOVE 'OFF' TO WS-READ-END-OF-FILE-SWITCH
    GO TO 1450-PARA-EXIT.
MOVE IM03-NAME TO WS-NAME.
```

In the example, a card will be READ each time the READ statement is executed. After the last card is READ and the statement is executed once more, 'OFF' will be MOVED to WS-READ-END-OF-FILE-SWITCH and control will go to the paragraph exit. If the AT END condition did not occur when the READ was executed, the MOVE statement following the period after the GO TO 1450-PARA-EXIT would be executed. If the period after the GO TO 1450-PARA-EXIT was accidentally omitted, the MOVE statement would be considered part of the AT END process and the statement <u>after</u> the MOVE would not be executed following a good READ. Likewise, if the period was placed after the MOVE 'OFF' statement, then the GO TO 1450-PARA-EXIT would be executed for each good READ. Clearly, placement of period after AT END is important to program execution.

Army Standard: (1) Only one READ per file is normally required. This should be in a one statement paragraph that is performed from the needed places in the program.

(2) Never execute a READ for a file which has encountered end-of-file unless it has been closed and reopened as input.

<u>WRITE</u>

A form of the WRITE verb is:

```
WRITE file-record [FROM record-name]  BEFORE  ADVANCING  data-name      LINES.
                                      AFTER              integer
                                                         mnemonic-name
```

This form writes to the printer or card-punch. Each time the WRITE statement is executed, the information in the record under the file description is written to the device specified (either a card punch or line printer). If the FROM option is included, the information in the record-name is copied to the file-record under the file description and then written. If the punch is being written to, then the BEFORE option should be chosen if the programer wants to determine which punch pocket the cards should be placed in after it is physically punched. A mnemonic name associated with the punch pocket in the SPECIAL-NAMES paragraph is used to select the actual pocket. If a printer is being written to, either BEFORE or AFTER may be used and the corresponding writing will take place. If BEFORE is specified, then the line being written will be written before the spacing. If AFTER is specified, then the line will be written after the spacing. Spacing is controlled by the ADVANCING clause. If a data-name is used, then it must be a non-negative elementary numeric item (normally a 77 entry) whose value lies between 0 and 99. If an integer is chosen, the same constraints on the values apply as with the data-name. If a mnemonic-name is chosen, then the mnemonic-name used must be defined in the SPECIAL-NAMES paragraph and associated with one of the printer carriage control channels.

EXAMPLES:

1. Write to the card punch:

```
SPECIAL-NAMES.
    S01 IS TO-POCKET-1.
    .
    .
    WRITE OP01-CARD-PUNCH BEFORE ADVANCING TO-POCKET-1.
```

or

```
    WRITE OP01-CARD-PUNCH.
```

2. Write to the printer.

```
SPECIAL-NAMES.
    C01 IS CHAN1.
    .
    .
    WRITE OR01-PRINTER-RCD AFTER ADVANCING CHAN1.
```

or

```
    WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE AFTER ADVANCING 2 LINES.
```

or

```
77  WS-CARR-CONTROL              PIC 99
                          VALUE 4.
    .
    .
    WRITE OR01-PRINTER-RCD BEFORE ADVANCING WS-CARR-CONTROL LINES.
```

In the first example, TO-POCKET-1 is associated with pocket one (S01) in the card punch. The write statement forces the punched card to be placed into pocket 1 through use of the BEFORE ADVANCING TO-POCKET-1 clause. In the second case of writing to the card punch, the programer simply allows the system to choose the pocket for the card to be placed in. Pocket selection is normally done when two or more types of cards are being punched and are to be kept separate. When the ADVANCING option is used, an extra character at the left of the record must be reserved for the system to use; e.g., to punch an 80 character record, 81 characters must be defined with the first character being a FILLER. When the BEFORE ADVANCING option is not used, then only 80 characters must be defined.

In the second example, CHAN1 is associated with printer carriage channel 1. Printer carriage channel 1 is almost always lined up with line six of the printer page. When, as in the first WRITE statement, the line is written AFTER ADVANCING CHAN1, the paper in the printer is advanced until printer carriage channel 1 is sensed then the line is written. This allows the programer to begin headings at the top of a new page. In the second case, the information in WS-DETAIL-LINE is moved to OR01-PRINTER-RCD then OR01-PRINTER-RCD is written after double spacing past the last line. In the third case, a data item is used to determine the spacing. The value that the printer is advanced BEFORE or AFTER writing the line depending on the option chosen. As in the card punch, whenever the ADVANCING option is used, an extra character on the left of the line must be reserved for system use. The IBM 1403 printer has 132 printable positions. When ADVANCING is used on this device, 133 characters must be written. The first character of the 133 is used by the system and the remaining 132 are printed.

Another form of the WRITE statement is:

WRITE file-record [FROM record-name].

This form is used to write to tape files. There is no BEFORE or AFTER option. The FROM clause is optional.

EXAMPLES:

WRITE OT01-TAPE-FILE-RCD.

WRITE OT01-TAPE-FILE-RCD FROM WS-ADD-RECORD.

In the first example, all of the information to be written to the tape file had been moved into OT01-TAPE-FILE-RCD (the 01 record description under the FD for the tape file) prior to the WRITE. In the second case, the information had been moved from WS-ADD-RECORD to OT01-TAPE-FILE RCD prior to WRITE. WS-ADD-RECORD would normally be defined in the WORKING-STORAGE SECTION somewhere.

The last form of the WRITE statement is:

WRITE file-record [FROM record-name]

INVALID KEY a series of COBOL verbs and associated parts except IF-ELSE (period).

This form is used to write to disk files. File-record is the 01 under the FD for the disk file. The FROM record-name is optional and is used identically as the other two examples. INVALID KEY is used to tell the programer when the disk file is full and no more records can be written to it. Disks files have a definite amount of space to hold the information they are given. When that space is all used, then the program has to be informed of it. This is done via the INVALID KEY clause. When the file is full and one more record is written, the series of COBOL verbs and associated parts after the INVALID KEY is executed. An IF verb cannot be one of the verbs used in the series.

Army Standard: (1) Only one WRITE for each record of a different size should be in a program. Each WRITE should be a one statement paragraph that is PERFORMed from the appropriate places.

(2) For files with several types of records, it is best to define them in WORKING-STORAGE and move them to the record area just before the WRITE.

DISPLAY

DISPLAY is a useful verb for debugging the program. It should be used to print intermediate results and values during the run of a program. It relieves the programer of defining a printer file during the debugging process and then having to pull all of the cards defining that file out of the program after the debugging phase is over. Its form is:

DISPLAY a series of literals (things enclosed in quotes) and data-names.

If the series is over 120 characters in length, only 120 characters will be printed on a line and the rest will be printed on the next line.

EXAMPLE:

```
DISPLAY 'NEW MASTER RECORD IS: ' OM01-NEW-MASTER-RECORD.
```

This example will print the literal 'NEW MASTER RECORD IS: ' followed by the characters in the OM01-NEW-MASTER-RECORD all on one line. Because the literal is 22 characters long, the number of characters in OM01-NEW-MASTER-RECORD that can be printed on the printer is 98. If OM01-NEW-MASTER-RECORD is longer than 98 characters, then only the first 98 characters are printed on this line and the rest are printed on the next line. Each time a DISPLAY is executed, the contents of the series of literals and variables are begun on a new line single spaced after the last line. There is no way to go to the top of a page with a DISPLAY statement. Double spacing can be achieved by simply displaying a blank before or after each real DISPLAY.

EXAMPLE:

```
DISPLAY ' '.
DISPLAY 'TOTAL NUMBER OF PEOPLE IS: ' WS-PEOPLE-TOTAL.
```

This example prints a blank line then a line giving the total people message.

Army Standard: Display should only be used as a debugging tool except in communications with the operator.

EXHIBIT (IBM Extension)

The EXHIBIT verb is another debugging tool. Its form is:

```
EXHIBIT  NAMED    series of data-names and literals.
         CHANGED
```

This verb formats each of the data-names into a DATA-NAME=value, form. These are spread out across the page. If the list is too long to fit on one line, then it is carried over to the next line. When the EXHIBIT NAMED form is used, then every data-name in the list is listed each time the EXHIBIT is excuted. When the EXHIBIT CHANGED form is used, all of them are reported the first time the verb is executed. The next time the verb is executed, only those data-names that have changed since the last execution are listed.

Army Standard: EXHIBIT should only be used as a debugging tool.

EXAMPLE:

```
EXHIBIT NAMED WS-DIVISION-HOLD WS-LINE-COUNT.

EXHIBIT CHANGED WS-DIVISION-HOLD WS-LINE-COUNT.
```

In the first example, WS-DIVISION-HOLD and WS-LINE-COUNT will be listed each time the statement is executed. In the second example, both WS-DIVISION-HOLD and WS-LINE-COUNT are printed the first time the EXHIBIT is executed. From then on whenever the statement is executed, only the data item that has changed (one or both) will be printed. If neither one has changed, then nothing is printed.

OPEN

The OPEN verb is used to prepare the files for input or output. Execution of this verb must precede the execution of any READ or WRITE statement. Its form is:

```
OPEN  INPUT    one or more file-names to be opened as either INPUT or OUTPUT
      OUTPUT

       INPUT    one or more file-names to be opened as either INPUT or OUTPUT...]
       OUTPUT
```

EXAMPLES:

```
OPEN INPUT PAY-MASTER-FILE.
OPEN INPUT CARD-READER DISK-FILE
    OUTPUT PRINTER NEW-DISK-FILE.
OPEN OUTPUT CARD-PUNCH PRINT-FILE.
```

The first example opens the PAY-MASTER-FILE as an input file and prepares that file for processing. The second example opens the CARD-READER and DISK-FILE as input files then opens the PRINTER and NEW-DISK-FILE as output files. The third example opens the CARD-PUNCH and PRINT-FILE as output files. Care should be exercised not to open a file more than once unless a CLOSE (described next) for that file has been executed in the interim. It is completely legal and necessary sometimes to OPEN a file as output then when processing is complete on that file, to close it and open it as input. All of the file names must be described in an FD in the FILE SECTION.

Army Standard: As many files as possible should be opened by one OPEN statement.

CLOSE

The CLOSE verb is used to disassociate a file with a program. Before the program is finished execution, all of the files currently open should be closed. If it is necessary to access a file as both input and output at different times, a CLOSE statement must be executed between the two types of access. The form of the CLOSE verb is:

CLOSE one or more file-names

EXAMPLES:

```
CLOSE INPUT-MASTER PRINTER.
CLOSE PRINTER.
CLOSE CARD-READER
    DISK-FILE.
```

All of the examples simply CLOSE the list of files after the CLOSE verb.

Army Standard: As many files as possible should be closed by one CLOSE statement.

LOGIC FLOW CONTROL VERBS

Paragraph names, while not verbs, are used by the logic control verbs in branching. They are A margin entries and are used to identify logically significant points in the program. They follow the same naming convention as data names; however, they may be completely numeric also. This is not recommended. They are ended with a period.

EXAMPLES:

```
0170-READ-AND-PRINT.
    .
    .
    .
0170-RAP-EXIT.
```

IF is used to make decisions during the execution of the program. Its form is:

IF condition any series of legal COBOL verbs and associated parts.

EXAMPLE:

```
IF IM01-SOC-SEC-NUM EQUAL TO IT01-SOC-SEC-NUM
    PERFORM 0100-UPDATE-ROUTINE THRU 0100-UPDATE-EXIT
    PERFORM 0200-READ-NEW-MASTER THRU 0200-READ-EXIT
    PERFORM 0300-READ-UPDATE THRU 0300-READ-EXIT.
```

In the above example, if IM01-SOC-SEC-NUM is equal to IT01-SOC-SEC-NUM, then the 0100-UPDATE-ROUTINE is PERFORMed, the 0200-READ-NEW-MASTER routine is PERFORMed, then the 0300-READ-UPDATE routine is PERFORMed. The entire series is executed only if the condition is true. If the condition is not true, then the sentence after the period ending the IF is executed. It is important to note that the first period after the IF ends that IF statement.

EXAMPLE:

```
IF A IS LESS THAN 0
    ADD 1 TO X
    ADD 2 TO Y.
    ADD 3 TO Z.

IF A IS LESS THAN 0
    ADD 1 TO X.
    ADD 2 TO Y
    ADD 3 TO Z.
```

NOTE: Army naming convention is not used for this example for illustrative purposes.

In the first case, if A is less than zero, one is added to X and two is added to Y then three is added to Z. If A is not less than zero, the adds to X and Y are skipped, then three is added to Z. The period ends the control of the IF. When the condition after the IF is true, the statements under control of the IF (e.g., the statements between the condition and the period) are executed. When the condition is not true, the statements under control of the IF are skipped and execution begins with the first statement after the period.

In the second case, if A is less than zero, one is added to X, the two is added to Y, and three is added to Z. If A is not less than zero, the add to X is skipped, then two is added to Y and three is added to Z.

Periods are very important in the logical construction of an IF statement. The omission or addition of a period can radically effect the execution of a program.

EXAMPLE:

```
IF A IS LESS THAN B
    ADD 1 TO A-COUNT
    GO TO A-EXIT.
ADD 1 to B-COUNT.
```

or

```
IF A IS LESS THAN B
    ADD 1 TO A-COUNT.
    GO TO A-EXIT.
ADD 1 TO B-COUNT.
```

NOTE: Army naming conventions are not used in this example for illustrative purposes.

In the first case, the branch to A-EXIT is only taken if A is less than B. If A is not less than B, then the statement after the period ending the IF is executed (ADD 1 TO B-COUNT). In the second case, if A is less than B or not less than B, the branch to A-EXIT is executed since the period ending the IF statement is after the ADD 1 TO A-COUNT. B-COUNT will never be incremented since execution cannot reach that point.

Another form of the IF statement is:

<u>IF</u> condition any series of COBOL verbs with associated parts (no periods)

<u>ELSE</u> any series of COBOL verbs with associated parts.

EXAMPLE:

```
IF IM01-DIVISION NOT = WS-DIVISION-SAVE
    OR WS-LINE-COUNT GREATER THAN 45
    PERFORM 0150-HEADING-ROUTINE THRU 0150-HEADING-EXIT
    MOVE 1 TO WS-LINE-COUNT
ELSE
    ADD 1 TO WS-LINE-COUNT.
MOVE IM01-NAME TO WS-NAME.
```

In the above example, the first condition was made up of two parts (a compound condition) consisting of IM01-DIVISION NOT = WS-DIVISION-SAVE and WS-LINE-COUNT GREATER THAN 45. The OR between the two tests means that if either (or both) of the tests is true, then everything until the ELSE is to be executed in order. If both of the tests are false, then everything after the ELSE will be executed instead. Then control is given to the sentence after the period (a MOVE statement).

Another connector between tests is the word AND. AND indicates that both conditions have to be true for the statements under the IF to be executed. If they are not true (either or both) then execution is given to the statements after the ELSE if present. If there is no ELSE associated with the IF, then execution goes to the first statement after the period ending the IF.

EXAMPLE:

```
IF IM01-DIVISION = WS-DIVISION-SAVE
    AND IM01-SOC-SEC-NUM EQUAL TO IT101-SOC-SEC-NUM
    PERFORM 0101-UPDATE-ROUTINE THRU 0101-UPDATE-EXIT
ELSE IF IM01-DIVISION = WS-DIVISION-SAVE
    AND IM01-SOC-SEC-NUM GREATER THAN IT201-SOC-SEC-NUM
    PERFORM 0102-ADD-ROUTINE THRU 0102-ADD-EXIT.
MOVE IM01-NAME TO WS-NAME.
```

In the above example, the first IF has a compound condition consisting of whether IM01-DIVISION was EQUAL TO WS-DIVISION-SAVE AND also whether IM01-SOC-SEC-NUM was EQUAL TO IT101-SOC-SEC-NUM. If both tests were true, then 0101-UPDATE-ROUTINE is PERFORMed. If either or both tests were false, then the statements after the ELSE are executed. The second IF also is made up of two tests: whether IM01-DIVISION is EQUAL TO WS-DIVISION-SAVE AND whether IM01-SOC-SEC-NUM is GREATER THAN IT201-SOC-SEC-NUM. If both tests were true, then 0102-ADD-ROUTINE is PERFORMed. If either or both of the tests were false, then the MOVE statement after the period ending the entire IF is executed. Notice that everything from the first IF to the final period is all one sentence (one logical construction). If any period had been in the middle of the sentence, then the IF will have ended there. Periods are very important things in IF statements. Notice also that one can (and many times should) put another IF as one of the COBOL verbs whose execution depends on the trueness or falseness of the original IF. This is called nesting IF's. In the above example, the test for equality between OM01-DIVISION and WS-DIVISION-SAVE is made twice. An example of how to avoid this and still have the same logical construction is given below.

EXAMPLE:

```
IF IM01-DIVISION = WS-DIVISION-SAVE
    IF IM01-SOC-SEC-NUM = IT101-SOC-SEC-NUM
        PERFORM 0101-UPDATE-ROUTINE THRU 0101-UPDATE-EXIT
    ELSE
        IF IM01-SOC-SEC-NUM GREATER THAN IT201-SOC-SEC-NUM
            PERFORM 0102-ADD-ROUTINE THRU 0102-ADD-EXIT.
MOVE IM01-NAME TO WS-NAME.
```

In the above example, the first IF has only one test: whether IM01-DIVISION is EQUAL TO WS-DIVISION-SAVE. Everything else in the sentence is dependent on the trueness or falseness of that test. If it was false, then control is given to the MOVE statement after the period. If is was true, then the second IF is executed. If the second condition was true, then 0101-UPDATE-ROUTINE is PERFORMed. After 0101-UPDATE-ROUTINE is finished, control is given to the MOVE statement after the period. If the second condition is false, then the third IF is executed. If that condition is true, then 0102-ADD-ROUTINE is PERFORMed. After 0102-ADD-ROUTINE is finished, control is given to the MOVE statement after the period. Notice that the first and third IF's have no ELSE associated with them. The rule for matching IF's with ELSE's is that an ELSE is associated with the first possible IF above the ELSE.

This allows the programer to build very complex logical constructions with compound conditions and nested IF's. Tremendous care should be exercised when designing these constructions.

If necessary, parentheses should be used in compound conditions to assure evaluation in the proper order. If OR and AND are used in the same compound, then the AND test is evaluated first.

EXAMPLE:

```
IF A IS NOT GREATER THAN B
    OR A + B IS EQUAL TO C
    AND D IS POSITIVE
```

will be evaluated as though it were parenthesized as follows:

```
IF (A IS NOT GREATER THAN B)
    OR (((A + B) IS EQUAL TO C)
        AND (D IS POSITIVE))
```

NOTE: Army naming conventions are not used for these examples for illustrative purposes.

The use of parentheses never will hurt a program's efficiency and can only add to the clarity of what the programer is trying to do. Also when going from one computer to another manufacturers computer, the evaluation rules could conceivably differ. With parentheses, there is no doubt for either the compiler or another programer having to make changes to the program in the absence of the original programer.

If at some point inside the nested IF's the programer would like to simply exit from the entire construction and go to the sentence after the period, the COBOL reserved phrase NEXT SENTENCE may be used.

EXAMPLE:

```
IF condition-1
    IF condition-2
        IF condition-3
            NEXT SENTENCE
        ELSE
           IF condition-4
                 .
                 .
           ELSE NEXT SENTENCE
    ELSE
       IF condition-5
            .
            .
```

```
          ELSE

             .
             .
    ELSE
       NEXT SENTENCE.
```

In the above example, if conditions 1, 2, and 3 are true, then control is simply given to the sentence after the period (the periods inside the nested IF's in the example only indicate a series of COBOL verbs and associated parts). If conditions 1 and 2 are true and conditions 3 and 4 are false, then again control is simply passed to the sentence after the period. If condition 1 is false, then the associated ELSE means to execute the sentence after the period. The last ELSE NEXT SENTENCE (period) is superfluous and was included to show the point. If the ELSE was not there and condition 1 was false, control would be given to the sentence after the period anyway. Some programers consider it good practice to match all IF verbs with an ELSE even if the ELSE is superfluous as in this case. If NEXT SENTENCE is put immediately after an ELSE or the conditions following an IF, no other verb except ELSE may follow it. Putting NEXT SENTENCE after a series of verbs and associated parts simply tells the compiler to do what it would have done anyway.

EXAMPLE:

```
IF condition-1
         .
         .
    NEXT SENTENCE
ELSE
         .
         .
         .
```

In the above example, NEXT SENTENCE is unnecessary. Control will go to the next sentence anyway.

88 level items can be used in conjunction with an IF.

EXAMPLE:

```
03  WS-PAY                     PIC 9(6)V99.
    88 SMALL                   VALUE .00 THRU 100.00.
    .
    .
    .
IF SMALL
    PERFORM 0505-SMALL-PAY-PROCESSING THRU 0505-SMALL-EXIT
ELSE
    .
    .
    .
```

is equivalent to:

```
IF (WS-PAY EQUAL TO .00 OR WS-PAY GREATER THAN .00)
    AND (WS-PAY LESS THAN 100.00 OR WS-PAY EQUAL TO 100.00)
    PERFORM 0505-SMALL-PAY-PROCESSING THRU 0505-SMALL-EXIT
ELSE
    .
    .
    .
    .
```

Clearly it is more convenient to use the 88 level item and there is the added benefit of being able to change the ranges of things being tested for by simply changing one card. 88's also increase the readability of the program and are self documenting with properly chosen names.

Army Standard:

(1) Use of the nested IF other than by the original programer is prohibited.
(2) Compound conditions requiring use of both AND and OR are prohibited.
(3) NOT will not be used in compound conditions.
(4) Avoid:
  (a) Using the NUMERIC and ALPHABETIC test unnecessarily.
  (b) Comparing numberic items with different number of decimal places. Move one of the items to a holding field that has the same PICTURE instead.
  (c) Comparing non-numeric items with different size PICTUREs.
  (d) Comparing group items that have unused areas in them.
  (e) Using the phrase ELSE NEXT SENTENCE unless it is in a nested IF.
(5) Do not specify a GO TO in and IF that goes to some code then returns the statement after the IF. Use a PERFORM instead or incorporate the code into the IF itself.
(6) NEXT SENTENCE can often be avoided by reversing the condition in the IF (using a NOT).
(7) Don't test the same data item for the same value once you have determined what the value is.
(8) Try to test for ranges of values instead of specific values, when appropriate.
(9) Don't use the ALPHABETIC test. Use IF...NOT NUMERIC instead.

```
EXAMPLES:           More efficient                 Less efficient
-----------------------------------------------------------------------------
#1                  IF A EQUALS B                  IF A EQUALS B
                      MOVE X TO Y                    GO TO C.
                      PERFORM Z THRU Z-EXIT.    D. MOVE ...
                    MOVE ...                       .
                    .                              .
                    .                              .
                    .                           C. MOVE X TO Y.
                                                   PERFORM Z THRU Z-EXIT.
                                                   GO TO D.
-----------------------------------------------------------------------------
#2                  IF A NOT EQUAL TO B            IF A EQUALS B
                      MOVE C TO D.                   NEXT SENTENCE
                                                   ELSE MOVE C TO D.
-----------------------------------------------------------------------------
#3                  IF A EQUALS 1                  IF A EQUALS 1
                      MOVE X TO B                    MOVE X TO B.
                    ELSE                           IF A EQUALS 7
                      IF A EQUALS 7                  MOVE Y TO B.
                        MOVE Y TO B                IF A EQUALS 9
                    ELSE                             MOVE Z TO B.
                      IF A EQUALS 9
                        MOVE Z TO B.
-----------------------------------------------------------------------------
```

```
#4                  IF A LESS THAN 5                  IF A EQUALS 1
                      IF A LESS THAN 1                  PERFORM B THRU B-EXIT.
                        PERFORM C THRU C-EXIT         IF A EQUALS 2
                      ELSE                              PERFORM B THRU B-EXIT.
                        PERFORM B THRU B-EXIT         IF A EQUALS 3
                    ELSE                                PERFORM B THRU B-EXIT.
                      PERFORM C THRU C-EXIT           IF A EQUALS 4
                                                        PERFORM B THRU B-EXIT.
                                                      IF A LESS THAN 1
                                                        PERFORM C THRU C-EXIT.
                                                      IF A GREATER THAN 4
                                                        PERFORM C THRU C-EXIT.
```

PERFORM

The PERFORM statement has a number of forms each having its own application. They will be discussed in detail one at a time. The basic concept of a PERFORM is common for all of them. While writing a program, that program can always be broken up into several discrete parts which function independently of each other. The main control section of the program simply gives control to the appropriate part at the appropriate time through a series of PERFORM statements. When the part that has been given control is finished, control is given to next COBOL verb after the PERFORM statement.

EXAMPLE:

```
    PERFORM 0100-ADD-ROUTINE THRU 0100-ADD-EXIT.
    CLOSE ADD-FILE.
        .

        .
0100-ADD-RTN.
    MOVE IT01-ADD-RCD TO OM01-NEW-MASTER-RECORD.
    WRITE OM01-NEW-MASTER-RECORD.
0100-ADD-EXIT. EXIT.
0200-SUBTRACT-RTN.
```

In the above example, when control is passed to the PERFORM 0100-ADD-RTN THRU 0100-ADD-EXIT statement, the PERFORM statement passes control to 0100-ADD-RTN. IT01-ADD-RCD is moved to OM01-NEW-MASTER-RECORD and then OM01-NEW-MASTER-RECORD is written. As soon as the 0100-ADD-EXIT paragraph is executed, the COBOL verb EXIT tells the machine to transfer control back to the PERFORM which passed control to thee 0100-ADD-RTN. The PERFORM then passes control to the next COBOL verb after the PERFORM (CLOSE ADD-FILE). The PERFORM is really a 'go do this then come back when your finished' verb. A simpler version, but less preferable, is:

EXAMPLE:

```
    PERFORM 0100-ADD-RTN.
    CLOSE ADD-FILE
    .
    .

0100-ADD-RTN.
    MOVE ADD-RECORD TO NEW-MASTER-RECORD.
    WRITE NEW-MASTER-RECORD.
0020-SUBTRACT-RTN.
```

Army Standard: An EXIT paragraph should always be the last paragraph in a PERFORM statement.

In the above example, exactly the same thing happens as the last example. While before the EXIT was explicitly defined in the PERFORM statement and written at the end of the paragraph that was being PERFORMED, this time the EXIT is implied to be at the end of the ADD-RTN paragraph just before the next paragraph name (0200-SUBTRACT-RTN.) and after the last period at the end of the last sentence in ADD-RTN (WRITE NEW-MASTER-RECORD.). It is good programing procedure to always use an explicit EXIT and PERFORM a paragraph through that EXIT. This makes the program more readable and makes the job of a maintenance programer fixing the program at a later date much easier. The reason for this will become clearer during the discussion of the GO TO verb.

Another form of the PERFORM verb is:

```
PERFORM paragraph-name [THRU paragraph-exit]
    UNTIL condition.
```

This form will PERFORM the paragraph as many times as necessary to make the condition true. This form can make an indefinite loop if the condition is never true. An example of this would be if the condition was A = B AND A NOT = B. This can never be true so the paragraph will be executed forever or until the operator gets tired of looking at the program executing. Clearly, this is not a desirable thing to do. Care should be used to insure that the condition will be made true at some point in the processing. If the condition is true when the PERFORM is executed, then the paragraph is never executed at all and control passes to the next verb. The condition may be an 88 level item. See the IF statement discussion for what it means.

Army Standard: (1) Always use PERFORM...THRU.
(2) Always execute the EXIT paragraph to return to the PERFORM.

EXAMPLE:

```
01  WS-EOF-TABS.
    03 MASTER-EOF-SWITCH                PIC X VALUE '0'.
       88 MASTER-DONE         VALUE '1'.

    PERFORM 0010-PARA-1 THRU 0010-PARA-EXIT
        UNTIL MASTER-DONE.
```

This example will PERFORM 0010-PARA-1 THRU 0010-PARA-EXIT until '1' is moved to WS-MASTER-EOF-SWITCH. When this happens, the 88 under WS-MASTER-EOF-SWITCH becomes true and the UNTIL is satisfied.

```
PERFORM paragraph-name [THRU paragraph-exit]
    n TIMES.
```

This form of the PERFORM verb will PERFORM the paragraph a fixed number of times where n is an integer that specifies how many times the paragraph is to be executed. If n is a variable, then it must be an integer.

EXAMPLE:

```
77 PERFORM-COUNT                                PIC S9(4)
    .                                 USAGE IS COMP
    .                                 VALUE IS +20
                                      SYNC.
PERFORM 5001-PARA-1 THRU 5001-PARA-EXIT
    PERFORM-COUNT TIMES.
```

or

```
PERFORM 5001-PARA-1 THRU 5001-PARA-EXIT
    20 TIMES.
```

Both of the examples do the same thing. They will PERFORM 5001-PARA-1 through 5001-PARA-EXIT 20 times.

The last form of the PERFORM verb is:

```
PERFORM paragraph-name [THRU paragraph-exit]
    VARYING name-1 FROM value-1 BY value-2
    UNTIL condition.
```

This form will initialize name-1 (index-name-1 or numeric-data-item-1) to value-1 (integer-1, index-name-2, or numeric-data-item-2) and test the condition (a simple or compound condition). If the condition is false, the paragraph is PERFORMed. When the paragraph is done the first time, name-1 is incremented by value-2 (integer-2, index-name-3, or numeric-data-item-3) and the condition is tested again. This process of PERFORMing the paragraph, incrementing name-1 and testing the condition continues until the condition is true. If the condition is true when the PERFORM is executed the first time, then control immediately passes to the next verb and the paragraph is never performed.

EXAMPLE:

```
PERFORM 0100-PARA-1 THRU 0100-PARA-EXIT
    VARYING TABLE-INDEX FROM 1 BY 1
    UNTIL TABLE-INDEX IS GREATER THAN 25.
```

This example initializes TABLE-INDEX to 1, then tests whether TABLE-INDEX is greater than 25. If it is not so, 0100-PARA-1 is executed. When control finally reaches 0100-PARA-EXIT, TABLE-INDEX is incremented by 1 (giving 2) and the condition is again evaluated. TABLE-INDEX is still not greater than 25, so 0100-PARA-1 again gets control; however, TABLE-INDEX is now equal to 2. This type of PERFORM is very useful in searching tables for an entry. More of how to use this form will be discussed in the section on tables.

EXIT

EXIT is a verb that provides a way to terminate a paragraph being PERFORMed and then return to the PERFORM statement for further processing if necessary. It can be the only thing after the paragraph-name other than comments ('*' in column 7) until the next paragraph-name.

GO TO

The GO TO is the most abused single verb in COBOL. When an error is found in a program, the average programer immediately begins to think in terms of GOing somewhere else. This leads to many unpredictable results initially and usually has some hidden implications that may not show up until the next programer has to repair the program again. A study was done to try to determine the cause of COBOL logical errors. The result of this study was to find the GO TO guilty of the vast majority. In this context, this text will present the way a GO TO *may* be used and then place several large constraints on *how* to use it. The form of the GO TO is:

```
GO TO paragraph-name.
```

EXAMPLE:

```
GO TO 0100-PARA-1.
```

In this example, the sentence immediately following 0100-PARA-1 will be executed. Indiscriminate use of this type of statement can lead to 'spaghetti' logic that is hard to debug and repair at a later date. A much more restrictive use is:

EXAMPLE:

```
GO TO 0100-PARA-EXIT.
```

This is the only type of GO TO allowed in 'structured' programing. In a given paragraph, the programer can only go to the EXIT of that paragraph.

EXAMPLE:

```
0525-PARA-1.
    IF LINE-COUNT GREATER THAN 20
        GO TO 0525-PARA-EXIT.
        .
        .
0525-PARA-EXIT.  EXIT.
```

This type of limitation requires that each paragraph be entered via a PERFORM statement. Most paragraphs won't need a GO TO in them at all. However, at some future date, another programer may find it necessary to put one in. To keep the program compatable, the initial programer should always have an EXIT for each paragraph and PERFORM THRU that exit. If the PROGRAM was filled with a mixture of PERFORMs and PERFORM THRUs and suddenly another programer needed to change one of the PERFORMs to a PERFORM THRU, all of the PERFORMs for that paragraph would have to be changed, leading to a lot of work and possible source of new problems. If all of the paragraphs have EXITs, then the programer simply GO TOs the EXIT and makes no other changes to the program. This cuts down on maintenance time, a major cause of cost overrun on a project and also makes the original programer spend more time on the design of the program leading to more trouble free code. Some programers complain that COBOL is already too verbose and wordy and all of these restrictions make his job harder and longer. That may be true, but the real cost of a program is not in the writing of it; it is in the maintenance of it. By forcing the programer to conform to this type of design forces him to use routine (break the program up into pieces that do discrete, separate things). This makes the problem clearer and the logic simpler.

STOP

This verb is used to either temporarily halt a program or to terminate it altogether. The form of the STOP statement is:

```
STOP   RUN
       quoted-literal
```

EXAMPLES:

```
STOP RUN.
STOP 'MOUNT 2402-TEST-DATA-TAPE'
```

The first example terminates the program. The second example halts the program and types the literal on the console typewriter. The operator will then restart the program after appropriate action has been taken on the message typed.

Army Standard: (1) There should be only one STOP RUN in the program. This should be a single statement paragraph.
(2) All files must be closed before the STOP RUN is executed.

ON (IBM EXTENSION)

The ON verb allows the programer to specify when a series of COBOL verbs are to be executed. Its form is:

```
ON integer-1  [AND EVERY integer-2] [UNTIL integer-3]
     any series of COBOL verbs and associated parts except IF-ELSE
     NEXT SENTENCE
[{ELSE     }  any series of COBOL verbs including IF-ELSE]
[{OTHERWISE}  NEXT SENTENCE                             ].
```

This verb sets up a count-condition. If only integer-1 has been specified, then the count-condition will only be satisfied once. The count is incremented by one each time the ON statement is executed. If the count-condition is not satisfied, the series of COBOL verbs after the ELSE is executed. If no ELSE is specified and the count-condition is not satisfied, execution goes to the next sentence after the period terminating the ON statement.

Army Standard: ON should only be used during debugging.

EXAMPLE:

```
ON 1 MOVE ZEROS TO WS-LINE-COUNT
    PERFORM 5626-HEADING-ROUTINE THRU 5626-HEADING-EXIT.

ON 10 AND EVERY 10
    PERFORM 1000-SUB-TOTAL-ROUTINE THRU 1000-SUB-EXIT.

ON 1 UNTIL 1000
    READY TRACE
ELSE RESET TRACE.

ON 25 AND EVERY 25 UNTIL 975
    NEXT SENTENCE
ELSE PERFORM 0500-PROCESS-ROUTINE THRU 0500-PROCESS-EXIT
    PERFORM 0600-COUNT-ROUTINE THRU 0600-COUNT-EXIT.
```

The first example executed the MOVE and PERFORM the first time the statement is executed. The second example will execute the PERFORM every ten times the statement is executed. The third example initiates a TRACE (described next) the first 1000 times the statement is executed. After the 1000th time, the ELSE path is taken and the TRACE is terminated. The fourth example simply goes to the period after the statement on the 25th time and each subsequent 25th time until the ON statement has been executed 975 times. If the particular time the ON is being executed is not one of the 25th times or is past the 975 limit, then the two PERFORMs under the ELSE are executed. This statement works very similarly to the IF-ELSE statement where the condition is described by the ON.

DEBUGGING

Tracing is a powerful tool for the programer. The format of the TRACE statement is:

$\left\{\begin{array}{l}\underline{\text{READY}}\\ \underline{\text{RESET}}\end{array}\right\}$ $\underline{\text{TRACE}}$ (IBM EXTENSION)

EXAMPLES:

```
READY TRACE.
RESET TRACE.
```

When the first example is executed, the COBOL listing number (number to the extreme left on the listing) of the paragraphs will be listed on the printer as they are executed. Any lines the program writes on the printer will be interspersed at the appropriate places. This allows the programer to actually see the order that the paragraphs in his program were executed and to hopefully find out why the logic is going awry. The second example turns off the trace. If the trace is off when it is turned off again, nothing happens. Likewise if the trace is on when it is turned on again, nothing happens. The TRACE verb can generate a large amount of paper and should be used with discretion. Intermediate values can be printed using the DISPLAY verb or the EXHIBIT verb during the run while using the trace and a clearer picture of what is going on is made available. A powerful use of the TRACE verb is in conjunction with an IF.

Army Standard: READY TRACE and RESET TRACE is only to be used as a debugging aid.

EXAMPLE:

```
IF IM01-ACCOUNT-NUMBER = 9546251
    READY TRACE.
```

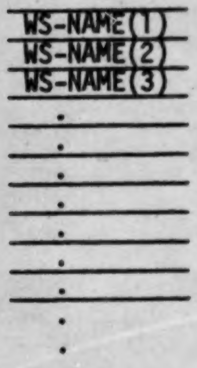
This is used to TRACE beginning at a problem account number. Presumably processing up to this point has gone correctly and no TRACE was necessary. This saves paper and processing time.

TABLE HANDLING

A table is defined in the DATA DIVISION via the OCCURS clause. It consists of a series of elements identical in form.

EXAMPLE: A table of twenty-five names:

looks like:



is defined by:

```
01  WS-NAME-TABLE.
    05  WS-NAME                            PIC X(20)
                                 OCCURS 25 TIMES.
```

This type of description is useful for storing a number of items that are the same size but differ in content. Once the description is complete, the individual item in the table may be referenced with either a subscript or index. A subscript or index simply tells the computer which one of the elements you want to look at or put something into.

EXAMPLE:

```
MOVE WS-NAME (1) TO WS-DETAIL-NAME.
```

The number in parentheses is a subscript. It indicates to COBOL that you are referencing the first of the twenty-five NAME entries. The thing in parentheses may also be a variable.

EXAMPLE:

```
77 WS-SUB                    PIC S9(2).
    .
    .
    .
MOVE 1 TO WS-SUB.
MOVE WS-NAME (WS-SUB) TO WS-DETAIL-NAME.
```

This example also moves the first of the twenty-five NAME entries to NAME-OUT. In this case, since a subscript is used, the contents of the subscript are examined and are used to indicate which of the NAME entries are to be operated on. When using a subscript, it is important not to let the subscript get larger than the maximum number of entires in the table. If a table must be examined from beginning to end looking for a particular entry, the subscript may be varied in increments of one.

EXAMPLE (LINEAR STYLE):

```
    MOVE 1 TO WS-SUB.
    MOVE '0' TO WS-FLAG.
A.  IF IM01-NAME EQUAL TO WS-NAME (WS-SUB)
        MOVE '1' TO WS-FLAG
        GO TO B.
    ADD 1 TO WS-SUB.
    IF WS-SUB LESS THAN 25
        GO TO A.
B.  ADD 1 TO WS-CARD-COUNT.
```

This example initializes WS-SUB to 1 then searches the name table for a match with IM01-NAME. If a match is found, WS-FLAG is set to 1. If a match is not found after all the names have been examined, WS-FLAG is set to 0. Another way to do the same thing is to use the PERFORM VARYING verb.

EXAMPLE (STRUCTURED STYLE):

```
MOVE '0' TO WS-FLAG
PERFORM 0100-NAME-COMPARE THRU 0100-NAME-EXIT
    VARYING WS-SUB FROM 1 BY 1
    UNTIL WS-SUB IS GREATER THAN 25
        OR WS-FLAG = '1'
    ADD 1 TO WS-CARD-COUNT.
    .
    .
    .
0100-NAME-COMPARE.
    IF IM01-NAME = WS-NAME (WS-SUB)
        MOVE '1' TO WS-FLAG.
0100-NAME-EXIT.  EXIT.
```

This example sets WS-FLAG to '0' then PERFORMs 0100-NAME-COMPARE through 0100-NAME-EXIT. During the process of the PERFORM, WS-SUB is set to 1 and the conditions are tested. If the conditions are not true, then 0100-NAME-COMPARE through 0100-NAME-EXIT are executed. After 0100-NAME-EXIT is finished, WS-SUB is incremented BY 1 and the conditions are tested. As soon as one of the conditions are true, the PERFORM statement is finished and the ADD 1 TO WS-CARD-COUNT is executed.

If the table consists of multiple entry items (e.g., name and social security number), the elementary items are referenced as though they are subscripted.

EXAMPLE:

```
01  WS-NAME-SSN-TABLE.
    05  WS-NAME-SSN                  OCCURS 25 TIMES.
        10  WS-NAME                  PIC X(20).
        10  WS-SSN                   PIC X(9).
    .
    .
    .
    IF WS-NAME (WS-SUB) EQUAL TO IM01-NAME
        MOVE '1' TO WS-FLAG.
```

Even though the description of WS-NAME does not contain an OCCURS clause, WS-NAME is subordinate to WS-NAME-SSN which does contain an OCCURS clause. This being the case, WS-NAME must be referenced as though WS-NAME itself contained the OCCURS. Another optional entry with the OCCURS clause is INDEXED BY. This clause specifies an index to be used with the particular table defined by the rest of the description.

EXAMPLE:

```
01  WS-BAS-TABLE.
    05  WS-BAS-ENTRY                      OCCURS 30 TIMES
                                          INDEXED BY BAS-INDEX.
        10  WS-BAS                        PIC S999V99.
        10  WS-RANK                       PIC X(3).
```

BAS-INDEX is the index associated with the BAS-ENTRY table. WS-BAS and WS-RANK are referenced exactly as in using a subscript.

EXAMPLE:

```
MOVE WS-BAS (BAS-INDEX) TO WS-DETAIL-BAS.
```

To initialize an index to a value and decrement or increment, the index, the SET verb, is used.

EXAMPLES:

```
SET BAS-INDEX TO 1.
SET BAS-INDEX UP BY 2.
SET BAS-INDEX DOWN BY 1.
```

Indexes are used in conditions exactly like subscripts.

EXAMPLE:

```
IF BAS-INDEX GREATER THAN 25
    .
    .
    .
```

The linear example given at the beginning of the table handling section would look like this, using indexing instead of subscription, assuming that INDEXED BY NAME-INDEX is specified after the OCCURS clause.

EXAMPLE:

```
    SET NAME-INDEX TO 1.
    MOVE '0' TO WS-FLAG.
A.  IF IM01-NAME EQUAL TO WS-NAME (NAME-INDEX)
        MOVE '1' TO WS-FLAG
        GO TO B.
    SET NAME-INDEX UP BY 1.
    IF NAME-INDEX LESS THAN 25
        GO TO A.
B.  ADD 1 TO WS-CARD-COUNT.
```

The structured example would look like:

```
    MOVE '0' TO WS-FLAG.
    PERFORM 0100-NAME-COMPARE THRU 0100-NAME-EXIT
        VARYING NAME-INDEX FROM 1 BY 1
        UNTIL NAME-INDEX IS GREATER THAN 25
            OR WS-FLAG = '1'.
    ADD 1 TO WS-CARD-COUNT.
        .
        .
        .
0100-NAME-COMPARE.
    IF IM01-NAME EQUAL TO WS-NAME (NAME-INDEX)
        MOVE '1' TO WS-FLAG.
0100-NAME-EXIT.  EXIT.
```

Indexing is usually an order of magnitude more efficient than subscripting when steping through a table from beginning to end. When using an index in a PERFORM VARYING statement, all of the SET statements are done automatically by the PERFORM.

88 entries cannot be used in or under an OCCURS.

Army Standard:
(1) Indexing is more efficient than subscripting and is perferred.
(2) Checks should be made to insure that indexes and subscripts do not exceed the range of the OCCURS.
(3) Subscripting or indexing must be limited to three levels.
(4) Subscripting or indexing beyond one level should be avoided.

DOS JOB CONTROL LANGUAGE (JCL)

// JOB name

This is the first card of each job. Name must be from 1 to 7 characters, the first of which must be alphabetic, the last of which may be characters or numbers. // is punched in columns 1 and 2.

// OPTION option-1, option-2, option-3...

This may appear after the JOB card and is used to change any of the standard system options. It is not required if the options which are standard are the options desired.

Options include:

LINK	which specifies that the job is to be written on disk for link editing.
LIST	which specifies that a listing of a compilation is to be written on SYSLST.
ERRS	specifies that a listing of any errors, if any, are to be listed on SYSLST.
DUMP	specifies that a hexadecimal dump of memory is to be made if the job is terminated abnormally.
LISTX	specifies that an ALC listing of generated instructions is to be written on SYSLST.
SYM	specifies that a table of data-names are to be listed with their corresponding system-assigned names with locations.
LOG	specifies that a listing of all job control is to be written on SYSLST.
XREF	specifies that a cross reference dictionary of data-names be written on SYSLST.
DECK	specifies that the compiler is to punch an object deck on SYSPCH.

The standard system options at the Institute of Administration are LIST, ERRS, NOLINK, NODUMP, NOLISTX, NOSYM, NOLOG, NOXREF, NODECK. Note that any option may be negated by preceding it with NO. If the programmer wanted to change one of the above options, such as NOLINK to LINK, he would punch the following card:

```
// OPTION LINK
```

If he wanted to change 2 items, such as NOLINK to LINK and LIST to NOLIST,

```
// OPTION LINK,NOLIST
```

NOTE: No space is permitted between a comma and the next option.

// ASSGN SYSnnn,X'cuu'

The assign clause is used to assign a system logical number to a specific hardward device.
nnn represents a 3 digit system logical number; cuu represents a channel and unit number to which a device is connected.

Devices include:

card reader	X'00C'
card punch	X'00D'
printer	X'00E'
tape drives	X'180' X'181' X'182' X'183' X'184' X'185'
disk drives	X'130' X'131' X'132' X'133' X'134' X'135' X'136
display (video)	X'020' X'021'
console typewriter	X'01F'

System logical numbers on our system are available from 000 thru 029.

// EXEC program-name

This statement causes a program on core-image library to be read into memory and executed. If no name is specified, the program compiled will be read into memory and executed.

EXAMPLE:

// EXEC LNKEDT
would cause the link editor to be read into memory and link edit the program just compiled.

// EXEC
would cause the program just link edited to be read into memory and executed.

/*

/* indicates end-of-data. It would appear after the last card of the source program to be compiled, for to the compiler the source program is data. It would also appear after any data which may have been included for use by an object program.

/&

/& indicates end-of-job. When this card is encountered, the job is terminated. The JOB card of the next job should follow a /& card.

EXAMPLE of a job to be compiled:

```
// JOB SAMPLE
// ASSGN SYS004,X'132'
// EXEC FCOBOL
       IDENTIFICATION DIVISION.
           PROGRAM-ID.  PE1.
           etc.
       rest of source program
/*
/&
```

NOTE: The full (ANSI) COBOL compiler requires a work file on disk, and expects SYS004 to be assigned to a disk. Our disk file used for this purpose is X'132', and thus the ASSGN card shown above is required to compile ANSI COBOL programs.

EXAMPLE of a job to be compiled and executed with data included:

```
// JOB EXAMPLE
// OPTION LINK
// ASSGN SYS004,X'132'
// EXEC FCOBOL
Source deck
/*
// EXEC LNKEDT
// ASSGN SYS005,X'00C'
// ASSGN SYS006,X'00E'
// EXEC
Card data
/*
/&
```

NOTE: Note that SYS005 must be assigned to a card reader in the COBOL source program since JCL has specified that SYS005 is to be assigned to X'00C, a card reader. Likewise, SYS006 must be assigned to a printer in the COBOL source program.

OS JOB CONTROL LANGUAGE (JCL)

There are three kinds of cards normally used in a typical OS job deck: the JOB card, the EXEC card, and the DD card. They will each be described to give an appreciation of the relationship to a COBOL program.

The JOB card has the following format:

```
//jobname JOB ... (other parameters)
```

This is always the first card in all JCL decks. It tells the operating system that a job is beginning.

The EXEC card has the following format when used in the context of a COBOL program being compiled, link-edited, and executed:

```
//         EXEC COBUCLG,PARM.COB=(compiler options),
//         PARM.LKED=(linkage-editor options),
//         COND.LKED=(n,LT),COND.GO=(n,LT)
```

The three cards shown, each beginning with a //, are treated as one card because the second and third cards are merely continuations of the first. A continuation is indicated by the card being continued ending with a comma and the next card beginning with a // followed by at least one space then the continued statement. In this case, the EXEC card is invoking a three step procedure (or PROC) that compiles a COBOL program, link-edits it, then executes that program. COBUCLG is the name of the procedure. The first step in the procedure is named the COB step. PARM.COB refers to options that affect the execution of this step. Some of the commonly used options are:

QUOTE APOST	indicates that a double quote is used to delineate literals instead of the apostophe (or single quote) or vice versa.
SPACE 1 SPACE 2 SPACE 3	indicate that the source program listing is to be single, double, or triple spaced respectively.
XREF NOXREF	indicates that a cross reference is to be generated or suppressed of all of the data-names and paragraph-names after the source program listing.
PMAP NOPMAP	indicates that a pseudo-assembler listing is to be printed or suppressed of the generated code in the PROCEDURE DIVISION after the source program listing and xref listing (if included).
DMAP NODMAP	indicates that a map of the DATA DIVISION is to be printed or suppressed after the source program listing.

If only one of the options is chosen, then the parentheses are not needed. If more than one option is chosen, the parentheses must be included and the options must be separated by commas.

The second step in the procedure is the LKED step. This is the step that link-edits the program. During this step, any subroutines that are used by the COBOL program are joined to it and a lot of busy work is done. There are two commonly used options in the linkage-editor.

MAP	indicates that a map of the linked program is to be printed. This is useful in debugging the program using a core dump.
XREF	indicates that a cross reference of the communication between the programs and subroutines is to be printed along with a map of the module.

The COND.LKED parameter is used to force execution of the LKED step even though diagnositcs (errors) above the level of W have been generated during the compilation of the COBOL program. Normally a program will execute with C level errors but not E level or higher. To allow the program to link and execute with a C level error, substitute an 8 for the n in this parameter and in the COND.GO parameter. If linking and k execution is desired only with W level errors, then omit both COND parameters and the comma after the parentheses on the last PARM parameter.

After the EXEC card is the SYSIN card for the COBOL program. Its format is:

```
//COB.SYSIN DD *
```

COB indicates that it is to be used in the COB step; SYSIN is the name of the DD card; DD indicates that it is indeed a DD card (data description); and * indicates that this is a card file with the cards immediately following the DD card. The COBOL program is placed immediately behind this card.

After the last card in the COBOL program, the DD cards that describe the files used in the COBOL program. Their format is:

```
//GO.ddname DD ... (additional parameters)
```

The GO indicates that it is used during the GO step (execution); the ddname is identical to the external name used in the system name after the ASSIGN in the FILE-CONTROL paragraph in the COBOL program; the DD indicates that this is a DD card; and the additional parameters can be many things. If this was to be a card file, then an asterisk (*) is all that is necessary after the DD. If this was a card punch file then SYSOUT=B is needed after the DD. If this was a printer file, then SYSOUT=A is needed after the DD. If the only entry after the DD is the word DUMMY and the file is used for sequential output (e.g., as a printer), everything written in that file is ignored by the system. If the DUMMY file is used as sequential input (e.g., as a card reader), the first read that is executed generates an end of file condition on that file. Random reads or writes are not allowed to DUMMY files.

If DISPLAY, EXHIBIT, or TRACE is used in the program, then the following card is needed.

```
//GO.SYSOUT DD SYSOUT=A
```

EXAMPLE:

```
//EXAMINE JOB ... (other JOB card parameters)
//        EXEC COBUCLG,
//             PARM.COB=(XREF,PMAP,DMAP,SPACE2),
//             PARM.LKED=XREF,
//             COND.LKED=(8,LT),
//             COND.GO=(8,LT)
//COB.SYSIN DD *
      IDENTIFICATION DIVISION.
      .
      .
      FILE-CONTROL.
          SELECT DISK-FILE
              ASSIGN TO UT-S-DISK.
          SELECT PRINTER
              ASSIGN TO UT-S-PRINTER.
          SELECT CARD-READER
              ASSIGN TO UT-S-READER.
          SELECT CARD-PUNCH
              ASSIGN TO UT-S-PUNCH.
      .
      .   (rest of COBOL program)
      .
//GO.DISK    DD ... (other DD card parameters)
//GO.PRINTER DD SYSOUT=A
//GO.READER DD *
.
.   (data cards)
.
//GO.PUNCH DD SYSOUT=B
//
```

In this example, the JOB name is EXAMPLE. The COBUCLG procedure is invoked. Parameters for the COB (COBOL compile) step are XREF, PMAP, DMAP, AND SPACE2. This will print a cross reference of all the data-names and paragraph names in the program, list all of the pseudo assembler generated codes in the procedure division, map the data division variables, and double space the COBOL source program listing. Parameters for the LKED (link-edit) step are XREF. This will map the linked module and cross reference the communication between the routines. The COND parameters indicate that linking and execution is desired with C level errors in compilation. The COB.SYSIN card indicates that the program immediately follows on cards.

The four files that the program uses are defined to have external names of DISK, PRINTER, READER, and PUNCH. These same names are all defined as DD names beneath the COBOL program. The GO.DISK DD card describes the disk file. The GO.PRINTER DD card indicates that this is a printer file by having SYSOUT=A after the DD. The GO.READER DD card indicates that this is a card file by the asterisk (*) after the DD. The GO.PUNCH DD card indicates that this is a card punch file by the SYSOUT=B after the DD. Finally the last card in the deck is a // with nothing else on the card. This indicates the end of the job.

COBOL SOURCE DESIGN LANGUAGE (SDL)

1. ADD: Adds the contents of one data name to another data name or a numeric integer value to a data name.

2. AT END: Used only in conjunction with the READ.

3. CLOSE: Makes a file nonaccessible, the file can no longer be read from or written to.

4. EXIT: Used to identify the ending point of a routine.

5. IF-THEN-ELSE: The IF portion states a condition, the THEN portion is done when condition is true, the ELSE portion is done when the condition is not true.

6. MOVE: Moves the contents of one data name to another data name.

7. OPEN: Makes a file accessible so that it can be read from or written to.

8. PERFORM: Does a routine from its beginning point through its ending point, one time.

9. PERFORM-UNTIL: Does a routine from its beginning point through its ending point, some number of times until a stated condition is true.

10. READ: Accesses a file and makes one record available for processing.

11. STOP RUN: Used to show the end of all processing. The STOP RUN will be used at the end of the main controlling routine.

12. WRITE: Puts the contents of a record name out to a physical device.

13. NEXT SENTENCE: Used in conjunction with the IF-THEN-ELSE to show a null THEN or ELSE portion of the statement.

230

ANS-COBOL Programming Standards

Software Division
Computer Science Department
US Army Institute of Administration
Ft Benjamin Harrison, Indiana 46216

February 1978

IA-01-02-05

The Programming Standards established here are not intended to be all-inclusive for ANS-COBOL program development. Rather, they are presented in order to eliminate ambiguity in the completion of COBOL practical exercises and tests by students of Computer Science Department. Standards already established by US Army Computer Systems Command are included in this memorandum. It should be noted that the standards specified are mandatory for graded practical exercises and tests administered by the Software Division. Specifications are grouped by COBOL Division, i.e., Section I deals with requirements in the Identification Division, Section II with Environment Division, Section III with Data Division and Section IV with Procedure Division. Where these standards conflict with ST 18-150, these standards have precedence.

Section I. IDENTIFICATION DIVISION

Though the only entry required by the COBOL compiler is that of a Program-ID, all paragraphs must be included in every program.

Requirements:

a. Program-ID. A meaningful name must be provided for this entry.

b. Author. The student will enter his Rank, Name and Class Number.

c. Installation: Enter

CSD, USAIA, Ft Harrison, IN 46216

d. Date-Written: The date the student started coding the program will be entered here.

e. Date-Compiled. Enter TODAY.

f. Security. Enter UNCLASSIFIED.

g. **Remarks.** This paragraph will be used to provide a description of the functions the program performs. It will also include an explanation of the use of all level 77 and all level 01 entries in working-storage. An asterisk (*) will be placed in column 7 of each line (card) of this paragraph, including the paragraph header line.

Example of Identification Division entries:

```
CONT A   B
7    8   12  16  20  24  28  32  36  40  44  48
 IDENTIFICATION DIVISION.
 PROGRAM-ID. D-STANDARD-EXAMPLE.
 AUTHOR. CPT I M A STUDENT, ADPC 3-77.
 INSTALLATION. CSD USAIA,FT HARRISON,IN46216.
 DATE-WRITTEN. 17 JAN 77.
 DATE-COMPILED. TODAY.
 SECURITY. UNCLASSIFIED.
*REMARKS.   THIS PROGRAM PRODUCES A LISTING
* OF PERSONNEL IN GRADE OF PV1 ASSIGNED TO
*    FORT HARRISON. EACH SOLDIER'S NAME, SSN,
*    AND UNIT OF ASSIGNMENT ARE PRINTED.
* DESCRIPTION OF 77'S.
***  WS-EOF-SW. THIS IS A 3 CHARACTER FIELD
*    INITIALIZED TO 'OFF'. IT IS USED TO
*    CONTROL THE READ-AND-PROCESS-LOOP. AT
*    END OF INPUT FILE ITS VALUE IS CHANGED
*    TO 'ON'.
* DESCRIPTION OF WORKING-STORAGE 01'S.
***  WS-HEADING-LINE. CONTAINS CONSTANT
*    DATA TO BE PRINTED AT TOP OF EVERY PAGE
*    OF THE LISTING.
***  WS-DETAIL-LINE. WILL CONTAIN DATA
*    PERTAINING TO EACH SOLDIER.
```

Section II. ENVIRONMENT DIVISION

Since student PE's are run on the IBM 360-40 under OS-MFT, the requirements for this section are oriented to that system.

a. In the Configuration Section both the Source-Computer and Object-Computer paragraphs will contain the entry: IBM-360-H40. The Special-Names paragraph will be used to equate the name "TOP-OF-PAGE" to the printer carriage control tape channel 1. Example of Configuration Section entries:

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-360-H40.
OBJECT-COMPUTER. IBM-360-H40.
SPECIAL-NAMES. C01 IS TOP-OF-PAGE.
```

b. Input-Output Section. The particular requirements for this section are contained in each practical exercise (PE), except for the formation of the individual file-name. The following standards will be used in forming file-names.

(1) The first character of the name must be I, indicating that the file is an input file in this program, or O, indicating that the file is an output file, or B, indicating that the file is used for both input and output. For sort files, the first character will be S.

(2) The second character will be selected from the following:

M - used as Master Files

T - used as Transaction Files

R - used as Report Files written to the printer

C - used for Card Files which contain miscellaneous data, e.g., data to build a table, control data to extract data from other files

W - Work (intermediate) Files

(3) The third character will always be a hyphen (-).

(4) The rest of the file-name must be descriptive of what the file contains.

234

(5) Examples:

(a) A personnel master file is used as input to a program

(b) A printed listing of personnel data is produced by a program. It is an output file.

(c) A supply parts master file, organized as an indexed sequential file, is read from the written to:

```
CONT A   B
7    8   12  16  20  24  28  32  36  40  44  48
    (a)  SELECT IM-PERS-MASTER
             ASSIGN TO UT-S-SYS006.
    (b)  SELECT OR-PERS-LISTING
             ASSIGN TO UT-S-SYS005.
    (c)  SELECT BM-PARTS
             ASSIGN TO DA-I-SYS006.
```

(6) The ASSIGN Clause of the SELECT statement will be indented four (4) columns and punched on a separate card.

Section III. DATA DIVISION

(a) File Section.

(1) Though the only clause required by the compiler in a File Description (FD) is the LABEL RECORD clause, all optional entries will be included for all files. Each clause will be punched on a separate card and will begin in column 12. Example of an FD:

```
FD  IM-PERS-MASTER
    LABEL RECORD IS STANDARD,
    BLOCK CONTAINS 1 RECORDS,
    RECORD CONTAINS 64 CHARACTERS,
    DATA RECORD IS IM01-PERS-MSTR.
```

The blocking factor and record size are specified on the record layouts for each PE. The data record name is a programmer-defined name and will be formed according to these standards:

(a) The first two characters will be identical to the first two characters of the file-name.

(b) The next two characters will consist of a sequence number beginning with 01. If there is only one record format associated with a file, then that record name will have 01 as the third and fourth characters. If there is more than one record format, the first one described will have a sequence number of 01, the second one 02, and so on.

(c) The fifth character will always be a hyphen (-).

(d) The rest of the data record name will be descriptive of the record itself.

(e) If a record is defined as an elementary item, the picture clause will begin in column 48 (or as specified in para b(4)) and the word PICTURE will be abbvreviated to PIC.

(2) Standards for defining data items within records are:

(a) Level numbers will begin at 05, and each subordinate level will be incremented by 5. Subordinate elements will be indented four (4) columns.

(b) The first five characters of all data item names will be identical to the first five characters of the record name. Allow two blank columns between level numbers and the data names.

(c) The rest of the data item name will be descriptive of the data item.

The following is an example of the description of the record associated with IM-PERS-MSTR, the file defined in paragraph a(l) of this section

```
01  IM01-PERS-MSTR.
    05  IM01-NAME.
        10  IM01-LAST-NAME              PIC X(12).
        10  IM01-FIRST-NAME             PIC X(8).
    05  IM01-SSN                        PIC 9(9).
    05  IM01-RANK                       PIC X(3).
    05  IM01-DATE-OF-RANK               PIC 9(6).
    05  IM01-NR-DEPS                    PIC 9(2).
    05  IM01-DATE-OF-BIRTH.
        10  IM01-DAY-OF-BIRTH           PIC 9(2).
        10  IM01-MON-OF-BIRTH           PIC 9(2).
        10   IM01-YR-OF-BIRTH           PIC 9(2).
    05  IM01-YTD-GROSS-PAY              PIC 9(5)V99
                    USAGE COMP-3.
    05  FILLER                          PIC X(14).
```

237

238

(d) The PICTURE clause for elementary items will begin in column 48 (or as specified in para b(4)). The word PICTURE will be abbreviated to PIC.

(e) Each additional clause will be punched on a separate card, and will begin in column 28.

b. Working-Storage Section.

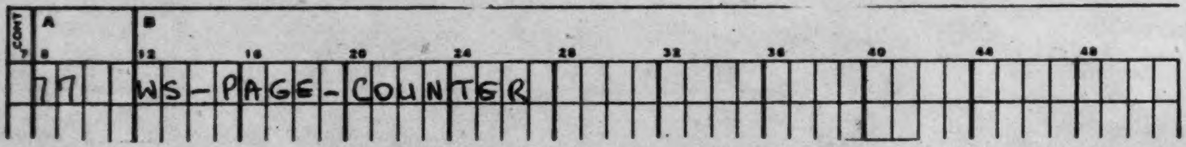
(1) The first entry in Working-Storage will be a level 77 item indicating the beginning of the section, and the last entry will be an 01 item indicating the end of the section. Insertion of these two items aids in debugging with a core dump. (See example on page 10.)

(2) All data elements defined in WORKING-STORAGE will be preceded by characters (WS-) as the first three characters of the name.

(3) The rest of the name will be formed as follows:

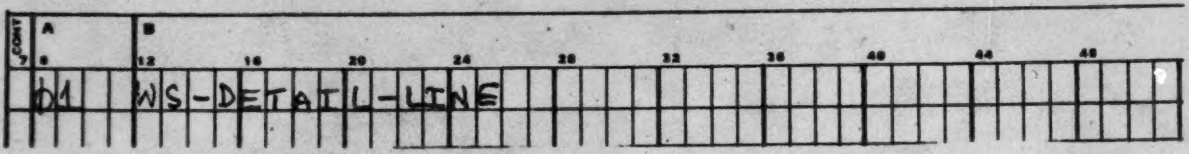
(a) For level 77 and 01 items, it is sufficient to provide a descriptive name. Examples:

To name a page counter using a level 77:


CONT
7
A
8
B
12
16
20
24
28
32
36
40
44
48

```
77  WS-PAGE-COUNTER
```

To name an independent record for a printed detail line using a level 01:


CONT
7
A
8
B
12
16
20
24
28
32
36
40
44
48

```
01  WS-DETAIL-LINE
```

(b) All data elements subordinate to a level 01 item will contain in positions 4 and 5 (and occasionally 6) of the name, an identifier which links them to the level 01 item: Examples: A field to be printed as part of a detail line and which contains a person's social security number could be named as: WS-HL1-SSN

A field to contain a date and be printed as part of Heading Line One could be named as: WS-HL1-DATE

(c) Rules for level numbers and indentation are the same as outlined for items in the file section. See para a(2)(a) of this section.

(4) Any clauses included with a date item description will be in the following order:

REDEFINES
OCCURS
PIC
USAGE
VALUE
JUSTIFIED
SYNC

(5) The first clause will be punched in column 48, and each additional clause will be punched on a separate card beginning in column 28.

(6) All numeric level 77 items will be assigned an initial value.

(7) The following example of a WORKING-STORAGE SECTION is provided to clarify the standards outlined above.

Working Storage EXAMPLE

```
WORKING-STORAGE SECTION.
77  WS-BEGIN-POINT                      PIC X(27)
                    VALUE 'WORKING STORAGE BEGINS HERE'.
77  WS-LINE-COUNTER                     PIC 9(2)
                    USAGE COMP
                    VALUE 21
                    SYNC.
77  WS-IM-EOF-SWITCH                    PIC X(3)
                    VALUE 'OFF'.

01  WS-DETAIL-LINE.
    05  FILLER                          PIC X(15)
                    VALUE SPACES.
    05  WS-DL-NAME.
        10  WS-DL-FIRST-NAME            PIC X(8).
        10  WS-DL-LAST-NAME             PIC X(12).
    05  FILLER                          PIC X(10)
                    VALUE SPACES.
    05  WS-DL-SSN                       PIC 9(9).
    05  FILLER                          PIC X(79)
                    VALUE SPACES.
01  WS-END-POINT                        PIC X(25)
                    VALUE 'WORKING-STORAGE ENDS HERE'.
```

Section IV. PROCEDURE DIVISION

a. **PARAGRAPH Names:**

(1) All paragraph names will contain in the first four positions a sequence number, beginning with 0010. The number assigned to each paragraph will be in ascending order and each will be incremented by 10 to allow insertion of additonal paragraphs as the need arises. An exception will be numbers assigned to EXIT paragraphs. These will contain the same number as the preceding paragraph. All paragraphs will have an associated exit paragraph, except, of course, exit paragraphs themselves.

(2) The fifth position of the name will be a hyphen (-), and the remainder of the name will be descriptive of the functions(s) performed.

(3) **Example:**

```
PROCEDURE DIVISION.
0010-MAIN-DRIVER.
    OPEN -----
    PERFORM 0020-READ-PRINT TRU 0020-EXIT
        UNTIL WS-IM-EOF-SW EQUAL 'ON'.
    CLOSE -----

0010-EXIT
    STOP RUN.

0020-READ-PRINT.
    READ -----
    MOVE -----
    WRITE -----

0020-EXIT
    EXIT.
```

b. **Verbs:** Any reference to indentation implies four (4) columns.

242

(1) PERFORM. The "Thru" option will <u>always</u> be used with a PERFORM verb. Procedure - name - 2, which follows the word THRU, must be an exit paragraph. Example:

```
       PERFORM 0030-READ-PRINT THRU 0030-EXIT.
```

```
   0030-READ-PRINT.
       READ IM-PERS-MASTER
           AT END
               MOVE 'ON' TO WS-IM-EOF-SWITCH.
       MOVE - - - - - - -
       WRITE - - - - - -
   0030-EXIT.
       EXIT.
```

(a) When the "UNTIL" option is used with a PERFORM, the UNTIL clause will be indented and appear on a separate line. Example:

```
       PERFORM 0030-READ-PRINT THRU 0030-EXIT
           UNTIL WS-IM-EOF-SW EQUAL 'ON'.
```

(b) When using the "VARYING" and "UNTIL" options together, each clause will be indented and appear on a separate line from the PERFORM. Example:

```
       PERFORM 0090-TABLE-BUILD THRU 0090-EXIT
           VARYING WS-SUB FROM 1 BY 1
           UNTIL WS-SUB GREATER 36.
```

(2) IF.

(a) All IF statements will include the optional word THEN following the condition. The word THEN will be indented and appear on the next line. If the statement following the THEN is another IF (i.e., a nested IF), it will also be indented and appear on a separate line. The same rules apply to the use of the ELSE option.

(b) Examples: See Page 14.

(3) Go To. A GO TO statement may only be used to transfer control to a section exit or a paragraph exit.

(4) SORT. All clauses specified with the SORT verb will be indented and each clause will appear on a separate line. When either or both of the "INPUT PROCEDURE IS" and "OUTPUT PROCEDURE IS" options are used, each of the sections referred to must be clearly identifiable as a separate section. A work section will be used as the last section of the program, and it will contain individual routines (with appropriate exists) which are referenced thru the use of PERFORM statements in the INPUT PROCEDURE section or OUTPUT PROCEDURE section. Example: See Page 15.

(5) READ. AT END and INVALID KEY conditions associated with READ statements will be indented and appear on a separate line. Only one READ statement will be used for each input file.

244

IF Statement EXAMPLES

```
     IF IM01-YEAR-OF-BIRTH LESS THAN 48
         THEN PERFORM 0080-PT-TEST-ROSTER THRU 0080-EXIT.




     IF  IM01-YEAR-OF-BIRTH LESS THAN 48
         THEN
             IF IM01-MARITAL-STATUS EQUAL 'N'
                 THEN PERFORM 0040-PRINT THRU 0040-EXIT
                 ELSE PERFORM 0120-ADD THRU 0120-EXIT

         ELSE
             IF IM01-PT-SCORE LESS THAN 300
                 THEN PERFORM 0110-PT-FAIL THRU 0110-EXIT
                 ELSE GO TO 0060-EXIT.

* NOTE: CORRESPONDING "THENS" AND "ELSE'S" ARE ALIGNED.
```

SORT Procedure EXAMPLES

```
           SORT SW-SORT-WORK-FILE
               ON ASCENDING KEY SW01-PART-NUMBER
               INPUT PROCEDURE IS 0020-IN-TO-SORT
               OUTPUT PROCEDURE IS 0030-SORT-OUT.



       0020-IN-TO-SORT SECTION.
           OPEN INPUT IM-PARTS-MASTER.
           PERFORM 0110-READ-RELEASE THRU 0110-EXIT
               UNTIL IM-EOF-SW EQUAL 'ON'.
           CLOSE IM-PARTS-MASTER.
       0020-SECTION-EXIT.
           EXIT.
       0030-SORT-OUT SECTION.
           OPEN OUTPUT OR-LISTING
           PERFORM 0120-RETURN-PRINT THRU 0120-EXIT
               UNTIL WS-SW-EOF-SWITCH EQUAL 'ON'.
           CLOSE OR-LISTING.
       0030-SECTION-EXIT.
           EXIT.
       0040-WORK SECTION.
       0110-READ-RELEASE.
           READ IM-PARTS-MASTER
               AT END ------
           ---- - -
       0010-EXIT.
           EXIT.
       0120-RETURN-PRINT.
           - - - - - - -
           - - - ---- -
       0120-EXIT.
           EXIT.
```

(6) WRITE. Only one WRITE statement per record type is permitted. When writing to the printer, both the FROM and AFTER ADVANCING options may be used. The AFTER ADVANCING clause will be indented and appear on a separate line.

(7) The following verbs will not be used:

MOVE Corresponding
ALTER
GO TO DEPENDING ON
ENTER
ON

(8) Miscellaneous. All switches will be set up as three characters alphanumeric fields. The words "OFF" and "ON" will be used to change the status of a switch, or as a literal to check for a specific value in a switch.

COBOL SORT

1. INTRODUCTION.

a. ANSI COBOL, with its IBM extensions, has many advanced processing capabilities; that is, it has features that are not really needed to write most programs but can make it easier to write the programs, or make the programs themselves more flexible. Among those items which might be considered advanced processing are the following:

(1) The segmentation feature, which allows programs to be divided into segments.

(2) The source program library, which allows parts of programs to be saved in a library for later use.

(3) The debugging feature, which allows for a diagnostic printout while the program is executing.

(4) The report writer feature, which provides an alternate method of describing printed output files.

(5) The sort feature, which permits rearranging records of a file into any order.

(6) Index-sequential file handling, which allows processing of an indexed-sequential direct-access file.

b. We do not have the time nor space to discuss all these features, nor do all of them merit discussion. We will cover the two that seem to be used most often: the sort feature and index-sequential file handling. The others we will leave for you to explore on your own.

2. OBJECTIVE. You will learn the COBOL statements necessary to create, read sequentially, and update randomly an index-sequential file. You will also learn the statements and methods needed to use the sort feature. (You should note that index-sequential files are an IBM addition to the standard COBOL language. Although many other manufacturers also have adopted index-sequential files, the system with which you are working may not have this feature.) The two IBM operating systems in general used by the Army are DOS (Disk Operating System) and OS (Operating System). Where one or the other operating system has unique requirements, they are identified.

3. THE COBOL SORT FEATURE. The COBOL sort feature enables you to rearrange a file used by your program into any order you want. You can do the sorting as part of your program, instead of having to use a separate utility. If you have a file in social security number order, for example, you can resort it into alphabetical order by name, or by address or into whatever order you would like. To use the sort, we must set up a sort-file in our program. The purpose of a sort-file is to gather up all the records to be sorted; to hold them while the sort takes place, and to parcel the records out once they have been rearranged. The sort-file is essentially a work area into which the records to be sorted are placed. One method of sorting consists of three separate phases; each is described below.

a. Phase 1 - Input. The records to be sorted are read into the computer's memory, and "strings" of records are built and written to the sort-file. A "string" consists of a series of records in the desired sequence. As soon as the sequence is broken, a "string" is complete, and a new one is started. For example, let us say that records 4, 7, 8, 10, and 13 have been read and are put into a "string". The next record read is 9. The sequence has been broken, and a completed "string" of records 4, 7, 8, 10, 13 is now on the sort-file. A new "string" with record 9 as the first record is now started.

b. Phase 2 - Sort/Merge. After all records have been read and placed in "strings", two "strings" are brought back into memory at a time, and merged into one longer "string". This continues until only two "strings" are left.

c. Phase 3 - Output. The last two "strings" are merged when the records are passed back to memory or to an output device - whichever the program calls for.

Note: The number of "strings" which are merged at one time depends on what is known as the "merge order". The "merge order" is a function of the amount of memory available, number and type of input/output devices and number of work files to be used.

4. When using the COBOL SORT feature, input to and output from the SORT can be handled in one of several ways. First, let us look at what a programer can do in the input phase. Basically, he has two options. They are:

a. Copy every field of every record in the input file to the sort-file.

b. Copy only selected fields and/or records in the input file to the sort-file.

5. If option "a" is selected, the input file is copied in its entirety. This method of input is useful when a file is simply to be rearranged, and all records are required for output. If option "b" is selected, only those records which need to be sorted are passed to the sort file. Also, individual fields which are not needed on output can be dropped, thereby making the sort record smaller. Removal of unnecessary records and fields reduces the amount of data to be sorted, and allows for better use of available memory. The time required for the sort itself will decrease. This option is normally used when printing a report from a file and

a. the report must be in a different order than the file,

b. only certain records are to be included and

c. not all fields on the original record are needed on the report.

6. Output may be one of two options:

a. Copy the sorted file to a permanent file;

b. Read the sorted file as input and process it, one record at a time, like any other sequential input file.

Note: Option "a" is used when a permanent copy of the sorted file is desired and no reports or special processing are required. Option "b" is used when reports and/or special processing are required.

7. DESCRIBING THE SORT-FILE (Sort Work Area). Like any other file, we must have a SELECT statement for the sort-file. However, instead of having a corresponding FD in the Data Division, we must use a Sort Description (SD). Let us look at the SELECT first.

a. SELECT for a sort-file under IBM Disk Operating System (DOS) COBOL. The file must be either a tape or a disk file. It is divided into sections, each section on a different tape or disk area. The more sections into which the file is broken, the faster the sort will execute. If you are using disk as your work unit, you may have from one to eight sections. If you are using tape, you may have from three to nine. You may not mix disk with tapes. The choice of how many work units to use will depend mostly on how many tapes or disk areas are available. The format of the SELECT clause may be found in the IBM ANS COBOL Language Reference Manual. An example of a SELECT for a sort-file that will use four work units on disk is SELECT SORTING, ASSIGN TO 4 SYS001-UT-2314-S-SORTWK1. Note that the ASSIGN clause for the sort-file must specify SYS001. The unit may be either 2400 (tape) or 2314 or 2311 (disk). The -SORTWK1 parameter must be included when you are using standard labels. An example of a SELECT statement for a sort-file under OS is:

```
SELECT SORT-WORK-FILE  ASSIGN TO 4  UT-S-SORTWK1
```

b. The sort description. In a program that uses a sort, each ordinary file must have an FD and the sort-file must have an SD. The format for the SD can be found in the IBM ANS COBOL Language Reference Manual under Sort File Description Entry.

Note: The sort-file-name must match the one given in the SELECT for the sort file, and the BLOCK CONTAINS clause may NOT be coded or used with an SD.

8. SORTING THE FILE. The file is sorted by executing a SORT statement in the PROCEDURE DIVISION. Each time the SORT is executed, the entire file is sorted. Usually, the SORT will be executed once. The SORT statement provides information about all phases of the sorting operation. The format of the SORT statement is shown below. The sort-file itself is never opened or closed by the programer. Opening and closing of the sort-file is done automatically by the SORT verb. For the format of the SORT statement, refer to the IBM ANS COBOL Language Reference Manual. Let us see how each part of the SORT statement applies to a particular phase of the SORT operation.

a. Phase 1--Input. The INPUT PROCEDURE or USING clause (not both) determines how the sort-file will get the records that it is to sort.

(1) If the entire input file is to be sorted, you can use the USING file-name-2 option. This will cause the file-name-2 file to be copied onto the sort-file. You must not open, close, or read the file-name-2 file; copying is done automatically. (NOTE: file-name-2 cannot be open while the sort is executing.) When you code the USING, you do not have access to the records as they are being copied onto the sort-file. The record description under the SD must be identical in form to the record description under file-name-2.

(2) If you wish to manipulate the records from the input file before they are placed on the sort-file, you must use the INPUT PROCEDURE clause. By coding the INPUT PROCEDURE clause, you will cause the input phase to perform the section-name you give in the INPUT PROCEDURE clause.

(3) A section is made up of one or more paragraphs that are logically related. Every section is begun with a section name (a paragraph name followed by a space and the word SECTION). A section ends on the statement before the next section name or with the last card in the program, whichever occurs first. It is a good idea to have an exit paragraph (called section-exit in this text) at the end of each section. This allows the structured construction of the procedure division to be maintained.

(4) When the sort begins execution, the input procedure section is executed. The input procedure section is actually a self-contained program that:

(a) opens the file that has the records to be sorted;
(b) reads those records;
(c) processes them in whatever way is needed;
(d) passes the records to the sort file by using a RELEASE statement (explained later);
(e) closes the input file;
(f) and branches to the section exit.

Note: The input procedure section has its own driver paragraph (ending with a GO TO to the section-exit), any routines needed for processing the input records, and a section exit at the end of all the routines.

b. Phase 2--Sort/Merge. The SORT...ON...KEY portion of the sort statement controls the action of this phase. Once Phase 1 is finished, all the records that are to be sorted will be arranged in strings in the sort-work area. The SORT...ON...KEY will determine how the strings are to be merged. Rules on sort keys are listed below. If the file was in SSN order and we wish to sort it alphabetically by name, then NAME is the key on which we will be sorting. There can be more than one sort key; for example, we might want to sort the records by STATE and then by CITY within each STATE. In this case the STATE field would be the major sort key, and the CITY field would be the minor sort key. When you code the ON...KEY clauses, you must code the major key first, followed by the minor keys. For each key, you have the choice of sorting it in ASCENDING (lowest first, highest last) sequence or DESCENDING (highest first, lowest last) sequence. (An alphabetical sort would be ASCENDING - A is first, Z is last). Rules on keys on DOS and OS:

(1) The sort keys that you are sorting on must be subdivisions of the record of the sort-file; i.e., they must come under the 01 which belongs to the SD.

(2) The maximum number of keys is 12.

(3) The total length of the keys must not exceed 256 bytes.

(4) Keys cannot contain or be subordinate to an OCCURS clause.

(5) Keys cannot be located after an OCCURS clause.

(6) All keys must be defined in the first 4092 bytes of the sort record.

c. Phase 3--Output. Once Phase 2 is finished, the records are in a sort work area and can be retrieved in the order in which you want them. The means of retrieving them is specified by the GIVING or OUTPUT PROCEDURE clauses.

(1) The GIVING file-name-3 operates in a similar manner to the USING; i.e., the sort-file is copied to an output file. The file that receives the sorted records must not be opened by the programer when the sort is executed. File-name-3 will be opened and closed automatically. As with the USING, you do not have access to the records as they are copied from the sort-file to file-name-3. The record description of file-name-3 must be identical in form to the record description in the sort-file.

(2) By using the OUTPUT PROCEDURE clause, you may retrieve the sorted records one at a time and process them. Like the input procedure, the section-name you give will be performed. Within the section you can retrieve records from the sort-file by using a RETURN statement (discussed later in this text). The sort file must not be opened or closed. As in the input procedure section, the output procedure section is responsible for opening and closing the output file(s), processing the records returned from the sort, and branching to its section exit when all processing is complete. It should begin with a driver routine, be followed by all the routines needed to process the sorted records and end with its section-exit paragraph. Once the performance of the OUTPUT PROCEDURE is finished, phase 3 of the sort operation is done; and control is passed to the sentence following the sort verb.

9. THE RELEASE STATEMENT. The purpose of an INPUT PROCEDURE is to let you process the records before they are passed on to the sort-file. Once you have finished processing an input record in your INPUT PROCEDURE, you must use the RELEASE statement to place the record on the sort-file. The format of the RELEASE can be found in the IBM ANS COBOL Language Reference Manual. The RELEASE is the equivalent of a WRITE statement for use on sort-files. Even

the FROM option works in the same way as it does for a WRITE; i.e., if the FROM is used, the data-name indicated in the FROM option is moved to the sort-record area and then the sort-record is RELEASED. The RELEASE statement writes the contents of the sort-record onto the sort-file. It may only be used within the bounds of the section named in the INPUT PROCEDURE clause.

10. THE RETURN STATEMENT. The purpose of an OUTPUT PROCEDURE is to let you process the records after they have been sorted. The RETURN statement may only be used within the bounds of the section named in the OUTPUT PROCEDURE clause. If the RELEASE is the equivalent of a WRITE for a sort file, then the RETURN is the equivalent of a READ for a sort-file. The format of the RETURN is found in the IBM ANS COBOL Language Reference Manual. Where the RELEASE called for the name of the sort records, the RETURN calls for the name of the sort file. The INTO option and AT END clause operate in the same way as for a READ. If the INTO option is used, the record description of the output file must be identical in form to the record description of the sort file. The AT END condition will occur when all the records of the sort file have been returned. The statements between the words AT END and the period that ends the sentence will not be executed until the AT END condition occurs.

11. SORT-WORK SECTION. A WORK section following the input and output procedure sections is used to contain all the routines used in the program. The input and output procedures then PERFORM routines contained in the SORT-WORK SECTION. The SORT-WORK SECTION is used not only to preserve a structured program design, but also to define the end of the input and output procedure sections.

12. A sample program which uses the SORT verb is at Inclosure 1.

PRACTICAL EXERCISE

Personnel Listing

1. The Personnel Office has requested a listing of all personnel in the battalion who have completed more than 35 months of service.
2. You are provided with a report format and card layout.
3. The report will be generated from the personnel master file. The file is stored in card-image form on a 2314 disk file.
4. You will keep a count of the total number of people who have more than 35 months of service. In addition, only 20 lines are to be printed (assuming double spacing) on a page. If more than 20 lines need to be printed start over with a new page.

FORM LAYOUT

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890123456789

          NAME                    RANK        SSN         MON SER

    XXXXXXXXXXXXXXXXXX            XXX      XXXXXXXXX        XXX

    XXXXXXXXXXXXXXXXXX            XXX      XXXXXXXXX        XXX

    XXXXXXXXXXXXXXXXXX            XXX      XXXXXXXXX        XXX

    TOTAL NUMBER OF PERSONNEL WITH OVER 35 MONTHS SERVICE   XXX
```

PERSONNEL MASTER INPUT FILE

Record Position	Field Name
1 - 2	Battalion
3	Company
4 - 12	Social Security Number
13 - 30	Name
31 - 32	Pay Grade
33 - 35	Rank
36 - 41	ETS
42 - 47	Date of Entry
48 - 50	Months of Service
51 - 52	Number of Dependents
53 - 80	Not Used

254

DUMP DEBUG WORKSHEET

1. Hexadecimal Memory Location of Interrupt ________________
2. Load Point ________________
3. Relative Interrupt Address ________________
4. ALC Instruction Interrupted ________________
5. Corresponding COBOL Statement Number ________________
6. COBOL Statement ________________________________
7. COBOL Source Data Names
   - Receiving Operand ________________________
   - Sending Operand ________________________
8. Internal Data Name for
   - Receiving Operand ________________________
   - Sending Operand ________________________
9. Data Format of
   - Receiving Operand ________________________
   - Sending Operand ________________________
10. Length in bytes reserved by
    - Receiving Operand ________________
    - Sending Operand ________________
11. Machine Instruction Interrupted ________________
12. Operation Code of Machine Instruction ________________
13. Machine Instruction Format ________________
14. Length of the Machine Instruction in bytes ________________
15. How many Operand Lengths in Machine Instr ________________
16. What is/are the Length(s) ________________

17. Base Register for First Operand ____________

18. Base Address for First Operand ____________

19. Displacement for First Operand ____________

20. Absolute Address of First Operand ____________

21. Contents of Storage Allocated to First Operand ____________

22. Base Register for Second Operand ____________

23. Base Address for Second Opernad ____________

24. Displacement for Second Operand ____________

25. Absolute Address of Second Operand ____________

26. Contents of Storage Allocated to Second Operand ____________

IF TEMPORARY STORAGE IS REFERENCED IN THE ALC INSTRUCTION (TS= in LISTX) THEN FILL IN ITEMS 27 THRU 30.

27. Relative Address of TS ____________

28. Absolute Address of TS ____________

29. Internal Data Name of Item in Left Half of TS ____________

30. Internal Data Name of Item in Right Half of TS ____________

31. Origin of Data and /or Reason for Incorrect Format:

a. Input Record ______ b. Failure to Initialize ______

c. Failure to Generate ______

32. Corrective Action:

D E B U G

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 8

PROGRAM NAME: DUMP-PE

```
                                                   IBM DOS AMERICAN NATIONAL STANDARD COBOL          CBF CL3-6       28/08/78
      1

  CBL LIB
  00001              BASIS C320MP
  00002     000010 IDENTIFICATION DIVISION.
  00003     000020 PROGRAM-ID. DUMP-PE.
  00004     000030 AUTHOR. LT HUDGIN.
  00005     000040 REMARKS. PE DESIGNED FOR USING ALC TO DEBUG A COBOL PROGRAM.
  00006     000050 ENVIRONMENT DIVISION.
  00007     000060 CONFIGURATION SECTION.
  00008     000070 SPECIAL-NAMES.
  00009     000080     C01 IS CHAN1.
  00010     000090 INPUT-OUTPUT SECTION.
  00011     000100 FILE-CONTROL.
  00012     000110     SELECT PERSONNEL-MASTER-FILE
  00013     000120       ASSIGN TO SYS006-UT-2314-S.
  00014     000130     SELECT PRINTER
  00015     000140       ASSIGN TO SYS005-UR-1403-S.
  00016     000150 DATA DIVISION.
  00017     000160 FILE SECTION.
  00018     000170 FD  PERSONNEL-MASTER-FILE
  00019     000180     LABEL RECORDS STANDARD.
  00020     000190 01  IM01-PERSONNEL-MASTER-RCD.                          PIC X(9).
  00021     000200     05  IM01-SOC-SEC-NUM                                PIC X(19).
  00022     000210     05  IM01-NAME                                       PIC X(2).
  00023     000220     05  IM01-DEPARTMENT                                 PIC X(5).
  00024     000230     05  FILLER                                          PIC 9(2).
  00025     000240     05  IM01-NUMBER-DEPENDENTS                          PIC 9(4)V9(2).
  00026     000250     05  IM01-MONTHLY-PAY                                PIC X(38).
  00027     000260     05  FILLER
  00028     000270 FD  PRINTER
  00029     000280     LABEL RECORDS OMITTED.                              PIC X(133).
  00030     000290 01  OR01-PRINTER-RCD
  00031     000300 WORKING-STORAGE SECTION.                                PIC X.
  00032     000310 77  WS-MASTER-FILE-EOF-SWITCH                           PIC S9(3)
  00033     000320 77  WS-TOTAL-FAMILY-SIZE-SUM                      USAGE IS COMP-3.
  00034     000330                                                         PIC S9(5)V9(2)
  00035     000340 77  WS-TOTAL-NET-PAY-SUM                          USAGE IS COMP-3.
  00036     000350                                                         PIC X(2)
  00037     000360 77  WS-DEPARTMENT-HOLD                            VALUE IS SPACES.
  00038     000370                                                         PIC S9(4)V9(2)
  00039     000380 77  WS-FEDERAL-TAX-HOLD                           USAGE IS COMP-3.
  00040     000390                                                         PIC S9(4)V9(2)
  00041     000400 77  WS-FICA-TAX-HOLD                              USAGE IS COMP-3.
  00042     000410                                                         PIC S9(4)V9(2)
  00043     000420 77  WS-NET-PAY-HOLD                               USAGE IS COMP-3.
  00044     000430                                                         PIC S9(3)
  00045     000440 77  WS-FAMILY-SIZE-HOLD                           USAGE IS COMP-3.
  00046     000450                                                         PIC S9V9(2)
  00047     000460 77  WS-TAX-RATE-HOLD                              USAGE IS COMP-3.
  00048     000470                                                         PIC SV9(3)
  00049     000480 77  WS-FICA-TAX-RATE                              USAGE IS COMP-3.
  00050     000490                                                         PIC SV9(3)
  00051     000500 77  WS-0-DEPENDENTS-TAX-RATE                      USAGE IS COMP-3
  00052          10
```

257

```
00053   000520                                          VALUE IS +.2.
00054   000530 77  WS-1-4-DEPENDENTS-TAX-RATE                  PIC SV9(3)
00055   000540                                          USAGE IS COMP-3
00056   000550                                          VALUE IS +.15.
00057   000560 77  WS-5-8-DEPENDENTS-TAX-RATE                  PIC SV9(3)
00058   000570                                          USAGE IS COMP-3
00059   000580                                          VALUE IS +.12.
00060   000590 77  WS-9-12-DEPENDENTS-TAX-RATE                 PIC SV9(3)
00061   000600                                          USAGE IS COMP-3
00062   000610                                          VALUE IS +.08.
00063   000620 77  WS-13-99-DEPENDENTS-TAX-RATE                PIC SV9(3)
00064   000630                                          USAGE IS COMP-3
00065   000640                                          VALUE IS +.05.
00066   000650 77  WS-SWITCH-ON                                PIC X
00067   000660                                          VALUE IS '0'.
00068   000670 77  WS-SWITCH-OFF                               PIC X
00069   000680                                          VALUE IS '1'.
00070   000690 01  WS-COLUMN-HEADING.
00071   000700     05  FILLER                                  PIC X(14)
00072   000710                                          VALUE IS SPACES.
00073   000720     05  FILLER                                  PIC X(18)
00074   000730                                          VALUE IS 'SSN'.
00075   000740     05  FILLER                                  PIC X(15)
00076   000750                                          VALUE IS 'NAME'.
00077   000760     05  FILLER                                  PIC X(13)
00078   000770                                          VALUE IS 'MONTHLY PAY'.
00079   000780     05  FILLER                                  PIC X(11)
00080   000790                                          VALUE IS 'NO OF DEPS'.
00081   000800     05  FILLER                                  PIC X(10)
00082   000810                                          VALUE IS 'PAY AVE'.
00083   000820     05  FILLER                                  PIC X(11)
00084   000830                                          VALUE IS 'FED TAX'.
00085   000840     05  FILLER                                  PIC X(11)
00086   000850                                          VALUE IS 'FICA TAX'.
00087   000860     05  FILLER                                  PIC X(19)
00088   000870                                          VALUE IS 'NET-PAY'.
00089   000880 01  WS-DETAIL-LINE.
00090   000890     05  FILLER                                  PIC X(11)
00091   000900                                          VALUE SPACES.
00092   000910     05  WS-SOC-SEC-NUM                          PIC X(9).
00093   000920     05  FILLER                                  PIC X(5)
00094   000930                                          VALUE IS SPACES.
00095   000940     05  WS-NAME                                 PIC X(18).
00096   000950     05  FILLER                                  PIC X(6)
00097   000960                                          VALUE SPACES.
00098   000970     05  WS-MONTHLY-PAY                          PIC $$$$9.99.
00099   000980     05  FILLER                                  PIC X(6)
00100   000990                                          VALUE SPACES.
00101   001000     05  WS-NUMBER-DEPENDENTS                    PIC Z9.
00102   001010     05  FILLER                                  PIC X(6)
00103   001020                                          VALUE IS SPACES.
00104   001030     05  WS-PAY-AVERAGE                          PIC $$$$9.99.
00105   001040     05  FILLER                                  PIC X(2)
```

```
00106   001050                                  VALUE IS SPACES.
00107   001060     05  WS-FEDERAL-TAX                   PIC $$$$9.99.
00108   001070     05  FILLER                           PIC X(5)
00109   001080                                  VALUE IS SPACES.
00110   001090     05  WS-TAX-RATE                      PIC VZ9.
00111   001100     05  FILLER                           PIC X(7)
00112   001110                                  VALUE IS ' %'.
00113   001120     05  WS-FICA-TAX                      PIC $$$$9.99.
00114   001130     05  FILLER                           PIC X(3)
00115   001140                                  VALUE IS SPACES.
00116   001150     05  WS-NET-PAY                       PIC $$$$9.99.
00117   001160     05  FILLER                           PIC X(11)
00118   001170                                  VALUE IS SPACES.
00119   001180 01  WS-FOOTING-LINE.
00120   001190     05  FILLER                           PIC X(35)
00121   001200                                  VALUE IS SPACES.
00122   001210     05  FILLER                           PIC X(31)
00123   001220                                  VALUE IS
00124   001230         'DEPARTMENT TOTAL FAMILY SIZE'.
00125   001240     05  WS-TOTAL-FAMILY-SIZE             PIC ZZ9.
00126   001250     05  FILLER                           PIC X(21)
00127   001260                                  VALUE IS ' AND TOTAL NET PAY'.
00128   001270     05  WS-TOTAL-NET-PAY                 PIC $$$$9.99.
00129   001280     05  FILLER                           PIC X(34)
00130   001290                                  VALUE IS SPACES.
00131   001300 PROCEDURE DIVISION.
00132   001310     OPEN INPUT PERSONNEL-MASTER-FILE
00133   001320          OUTPUT PRINTER.
00134   001330     MOVE WS-SWITCH-OFF TO WS-MASTER-FILE-EOF-SWITCH.
00135   001340     PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT
00136   001350       UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.
00137   001360     CLOSE PERSONNEL-MASTER-FILE
00138   001370           PRINTER.
00139   001380   . STOP RUN.
00140   001390 0010-READ-AND-PRINT.
00141   001400     READ PERSONNEL-MASTER-FILE
00142   001410       AT END
00143   001420         PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
00144   001430         MOVE WS-SWITCH-ON TO WS-MASTER-FILE-EOF-SWITCH
00145   001440         GO TO 0010-EXIT.
00146   001450     IF WS-DEPARTMENT-HOLD EQUAL TO SPACES
00147   001460       PERFORM 0030-HEADING-RTN THRU 0030-EXIT
00148   001470     ELSE
00149   001480       IF WS-DEPARTMENT-HOLD NOT EQUAL TO IM01-DEPARTMENT
00150   001490         PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
00151   001500         PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
00152   001510     IF IM01-NUMBER-DEPENDENTS EQUAL TO ZEROS
00153   001520       MOVE WS-0-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00154   001530     ELSE
00155   001540       IF IM01-NUMBER-DEPENDENTS LESS THAN 5
00156   001550         MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00157   001560       ELSE
00158   001570         IF IM01-NUMBER-DEPENDENTS LESS THAN 9
```

```
00159   001580             MOVE WS-5-8-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00160   001590           ELSE
00161   001600             IF IM01-NUMBER-DEPENDENTS LESS THAN 13
00162   001610               MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00163   001620             ELSE
00164   001630               MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
00165   001640                 WS-TAX-RATE-HOLD.
00166   001650       MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
00167   001660         GIVING WS-FEDERAL-TAX-HOLD.
00168   001670       MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
00169   001680         GIVING WS-FICA-TAX-HOLD.
00170   001690       SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
00171   001700         IM01-MONTHLY-PAY
00172   001710         GIVING WS-NET-PAY-HOLD
00173   001720       DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
00174   001730         GIVING WS-PAY-AVERAGE.
00175   001740       MOVE IM01-SOC-SEC-NUM TO WS-SOC-SEC-NUM.
00176   001750       MOVE IM01-NAME TO WS-NAME.
00177   001760       MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
00178   001770       MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
00179   001780       MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
00180   001790       MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
00181   001800       MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
00182   001810       MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
00183   001820       ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
00184   001830       ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
00185   001840       WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
00186   001850         AFTER ADVANCING 2 LINES.
00187   001860 0010-EXIT. EXIT.
00188   001870 0020-TOTALS-RTN.
00189   001880       MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
00190   001890       MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
00191   001900       WRITE OR01-PRINTER-RCD FROM WS-FOOTING-LINE
00192   001910         AFTER ADVANCING 3 LINES.
00193   001920       MOVE ZEROS TO WS-TOTAL-FAMILY-SIZE-SUM
00194   001930         WS-TOTAL-NET-PAY-SUM.
00195   001940 0020-EXIT. EXIT.
00196   001950 0030-HEADING-RTN.
00197   001960       MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
00198   001970       WRITE OR01-PRINTER-RCD FROM WS-COLUMN-HEADING
00199   001980         AFTER ADVANCING CHAN1.
00200   001990 0030-EXIT. EXIT.
```

```
5

INTRNL NAME   LVL SOURCE NAME                           BASE    DISPL  INTRNL NAME   DEFINITION   USAGE       R  O  Q  M

DNM=1-151     FD  PERSONNEL-MASTER-FILE                 DTF=01         DNM=1-151                  DTFSD                F
DNM=1-196     01  IM01-PERSONNEL-MASTER-RCD              BL=1    000   DNM=1-196     DS 0CL80     GROUP
DNM=1-234     02  IM01-SOC-SEC-NUM                       BL=1    000   DNM=1-234     DS 9C        DISP
DNM=1-260     02  IM01-NAME                              BL=1    009   DNM=1-260     DS 18C       DISP
DNM=1-279     02  IM01-DEPARTMENT                        BL=1    019   DNM=1-279     DS 2C        DISP
DNM=1-304     02  FILLER                                 BL=1    01D   DNM=1-304     DS 5C        DISP
DNM=1-320     02  IM01-NUMBER-DEPENDENTS                 BL=1    022   DNM=1-320     DS 2C        DISP-NM
DNM=1-352     02  IM01-MONTHLY-PAY                       BL=1    024   DNM=1-352     DS 6C        DISP-NM
DNM=1-378     02  FILLER                                 BL=1    02A   DNM=1-378     DS 38C       DISP
DNM=1-397     FD  PRINTER                               DTF=02         DNM=1-397                  DTFPR                F
DNM=1-429     01  OR01-PRINTER-RCD                       BL=2    000   DNM=1-429     DS 133C      DISP
DNM=1-454     77  WS-MASTER-FILE-EOF-SWITCH              BL=3    000   DNM=1-454     DS 1C        DISP
DNM=2-000     77  WS-TOTAL-FAMILY-SIZE-SUM               BL=3    001   DNM=2-000     DS 2P        COMP-3
DNM=2-034     77  WS-TOTAL-NET-PAY-SUM                   BL=3    003   DNM=2-034     DS 4P        COMP-3
DNM=2-067     77  WS-DEPARTMENT-HOLD                     BL=3    007   DNM=2-067     DS 2C        DISP
DNM=2-095     77  WS-FEDERAL-TAX-HOLD                    BL=3    009   DNM=2-095     DS 4P        COMP-3
DNM=2-124     77  WS-FICA-TAX-HOLD                       BL=3    00D   DNM=2-124     DS 4P        COMP-3
DNM=2-150     77  WS-NET-PAY-HOLD                        BL=3    011   DNM=2-150     DS 4P        COMP-3
DNM=2-178     77  WS-FAMILY-SIZE-HOLD                    BL=3    015   DNM=2-178     DS 2P        COMP-3
DNM=2-207     77  WS-TAX-RATE-HOLD                       BL=3    017   DNM=2-207     DS 2P        COMP-3
DNM=2-233     77  WS-FICA-TAX-RATE                       BL=3    019   DNM=2-233     DS 2P        COMP-3
DNM=2-259     77  WS-0-DEPENDENTS-TAX-RATE               BL=3    01B   DNM=2-259     DS 2P        COMP-3
DNM=2-293     77  WS-1-4-DEPENDENTS-TAX-RATE             BL=3    01D   DNM=2-293     DS 2P        COMP-3
DNM=2-329     77  WS-5-8-DEPENDENTS-TAX-RATE             BL=3    01F   DNM=2-329     DS 2P        COMP-3
DNM=2-365     77  WS-9-12-DEPENDENTS-TAX-RATE            BL=3    021   DNM=2-365     DS 2P        COMP-3
DNM=2-402     77  WS-13-99-DEPENDENTS-TAX-RATE           BL=3    023   DNM=2-402     DS 2P        COMP-3
DNM=2-440     77  WS-SWITCH-ON                           BL=3    025   DNM=2-440     DS 1C        DISP
DNM=2-462     77  WS-SWITCH-OFF                          BL=3    026   DNM=2-462     DS 1C        DISP
DNM=3-000     01  WS-COLUMN-HEADING                      BL=3    028   DNM=3-000     DS 0CL122    GROUP
DNM=3-030     02  FILLER                                 BL=3    028   DNM=3-030     DS 14C       DISP
DNM=3-049     02  FILLER                                 BL=3    036   DNM=3-049     DS 18C       DISP
DNM=3-068     02  FILLER                                 BL=3    048   DNM=3-068     DS 15C       DISP
DNM=3-087     02  FILLER                                 BL=3    057   DNM=3-087     DS 13C       DISP
DNM=3-106     02  FILLER                                 BL=3    064   DNM=3-106     DS 11C       DISP
DNM=3-125     02  FILLER                                 BL=3    06F   DNM=3-125     DS 10C       DISP
DNM=3-144     02  FILLER                                 BL=3    079   DNM=3-144     DS 11C       DISP
DNM=3-163     02  FILLER                                 BL=3    084   DNM=3-163     DS 11C       DISP
DNM=3-182     02  FILLER                                 BL=3    08F   DNM=3-182     DS 19C       DISP
DNM=3-201     01  WS-DETAIL-LINE                         BL=3    0A8   DNM=3-201     DS 0CL133    GROUP
DNM=3-231     02  FILLER                                 BL=3    0A8   DNM=3-231     DS 11C       DISP
DNM=3-250     02  WS-SOC-SEC-NUM                         BL=3    0B3   DNM=3-250     DS 9C        DISP
DNM=3-274     02  FILLER                                 BL=3    0BC   DNM=3-274     DS 5C        DISP
DNM=3-293     02  WS-NAME                                BL=3    0C1   DNM=3-293     DS 18C       DISP
DNM=3-310     02  FILLER                                 BL=3    0D3   DNM=3-310     DS 6C        DISP
DNM=3-329     02  WS-MONTHLY-PAY                         BL=3    0D9   DNM=3-329     DS 8C        NM-EDIT
DNM=3-364     02  FILLER                                 BL=3    0E1   DNM=3-364     DS 6C        DISP
DNM=3-383     02  WS-NUMBER-DEPENDENTS                   BL=3    0E7   DNM=3-383     DS 2C        NM-EDIT
DNM=3-420     02  FILLER                                 BL=3    0E9   DNM=3-420     DS 6C        DISP
DNM=3-439     02  WS-PAY-AVERAGE                         BL=3    0EF   DNM=3-439     DS 8C        NM-EDIT
DNM=3-474     02  FILLER                                 BL=3    0F7   DNM=3-474     DS 2C        DISP
```

IA-02-03-15

6

7

```
             MEMORY MAP

                TGT                           00548

          SAVE AREA                           00548
          SWITCH                              CC550
          TALLY                               0C594
          SCFT SAVE                           CC558
          ENTRY-SAVE                          0055C
          SCRT CORE SIZE                      005A0
          NSTC-REELS                          CC5A4
          SCFT RET                            005A6
          WCRKING CELLS                       005A8
          SORT FILE SIZE                      0C6D8
          SORT MODE SIZE                      CC6DC
          PGT-VN TBL                          CC6E0
          TGT-VN TBL                          006E4
          SORTAB ADDRESS                      0C6E8
          LENGTH OF VN TBL                    0C6EC
          LNGTH OF SCRTAB                     0C6EE
          PGM ID                              CC6FC
          A(INIT1)                            006F8
          UPSI SWITCHES                       006FC
          CVERFLOW CELLS                      00704
          BL CELLS                            CC7C4
          CTFACR CELLS                        CC710
          TEMP STORAGE                        CC718
          TEMP STORAGE-2                      CC728
          TEMP STCRAGE-3                      00738
          TEMP STORAGE-4                      CC738
          BLL CELLS                           00738
          VLC CELLS                           0C73C
          SBL CELLS                           0C73C
          INDEX CELLS                         0C73C
          SUBADR CELLS                        0073C
          CNCTL CELLS                         0073C
          PFMCTL CELLS                        0073C
          PFMSAV CELLS                        0C73C
          VN CELLS                            00750
          SAVE AREA =2                        0075C
          XSASW CELLS                         0075C
          XSA CELLS                           0075C
          PARAM CELLS                         CC75C
          RPTSAV AREA                         0075C
          CHECKPT CTR                         0075C
          IOPTR CELLS                         0075C

LITERAL POOL (HEX)

C07DC (LIT+0)      C5C5C8CC1  3C4C2058  20202120  48202040  202120F0  20212000
007E0 (LIT+24)     0C0C00C0   0C00C004  5B58C2C4  07C5D540  5B58C2C3  D3D6E2C5
```

```
          PGT                         00768

        OVERFLOW CELLS                00768
        VIRTUAL CELLS                 00768
        PROCEDURE NAME CELLS          0076C
        GENERATED NAME CELLS          00784
        SUBDTF ADDRESS CELLS          007C4
        VNI CELLS                     007C4
        LITERALS                      007CC
        DISPLAY LITERALS              008C0

 REGISTER ASSIGNMENT

  REG 6    BL =3
  REG 7    BL =1
  REG 8    BL =2

132             000800                      START    EQU    *
                000800  58 10 D 1C8                  L      1,1C8(0,13)      DTF=1
                000804  48 10 C 086                  SH     1,086(0,12)      LIT+30
                000808  94 BF 1 000                  NI     000(1),X'BF'
                00080C  41 10 C 088                  LA     1,088(0,12)      LIT+32
                000810  58 00 D 1C8                  L      0,1C8(0,13)      DTF=1
                000814  18 40                        LR     4,0
                000816  05 F0                        BALR   15,0
                000818  50 00 F 008                  ST     0,008(0,15)
                00081C  45 00 F 00C                  BAL    0,00C(0,15)
                000820  00000000                     DC     X'00000000'
                000824  0A 02                        SVC    2
                000826  41 10 C 088                  LA     1,088(0,12)      LIT+32
                00082A  58 00 D 1CC                  L      0,1CC(0,13)      DTF=2
                00082E  18 40                        LR     4,0
                000830  07 00                        BCR    0,0
                000832  05 F0                        BALR   15,0
                000834  50 00 F 008                  ST     0,008(0,15)
                000838  45 00 F 00C                  BAL    0,00C(0,15)
                00083C  00000000                     DC     X'00000000'
                000840  0A 02                        SVC    2
                000842  50 20 D 1C0                  ST     2,1C0(0,13)      BL =2
                000846  58 80 D 1C0                  L      8,1C0(0,13)      BL =2
134             00084A  D2 00 6 000 6 026            MVC    000(1,6),026(6)  DNM=1-454     DNM=2-462
135             000850  58 00 D 208                  L      0,208(0,13)      VN=01
                000854  50 00 D 1F4                  ST     0,1F4(0,13)      PSV=1
                000858  58 00 C 01C                  L      0,01C(0,12)      GN=01
                00085C  50 00 D 208                  ST     0,208(0,13)      VN=01
                000860                      GN=01    EQU    *
                000860  58 20 C 020                  L      2,020(0,12)      GN=02
                000864  D5 00 6 000 6 025            CLC    000(1,6),025(6)  DNM=1-454     DNM=2-440
                00086A  07 82                        BCR    8,2
                00086C  58 10 C 004                  L      1,004(0,12)      PN=01
          0009D4   07 F1                               BCR    15,1
155       0009D6                             GN=011    EQU    *
          0009D6   F2 71 D 1DC 7 022                   PACK   1D0(8,13),022(2,7)     TS=01          DNM=1-320
          0009DC   F9 10 D 1D6 C 069                   CP     1D6(2,13),069(1,12)    TS=07          LIT+1
          0009E2   58 F0 C 04C                         L      15,04C(0,12)           GN=013
          0009E6   07 AF                               BCR    10,15
156       0009E8   F8 71 D 1DC 6 01D                   ZAP    1D0(8,13),01D(2,6)     TS=01          DNM=2-293
          0009EE   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          0009F4   F8 11 6 017 D 1D6                   ZAP    017(2,6),1D6(2,13)     DNM=2-207      TS=07
158       0009FA   58 10 C 048                         L      1,048(0,12)            GN=012
          0009FE   07 F1                               BCR    15,1
159       000A02                             GN=013    EQU    *
          000A02   F2 71 D 1DC 7 022                   PACK   1D0(8,13),022(2,7)     TS=01          DNM=1-320
          000A08   F9 10 D 1D6 C 06A                   CP     1D6(2,13),06A(1,12)    TS=07          LIT+2
          000A0C   58 F0 C 050                         L      15,050(0,12)           GN=014
          000A10   07 AF                               BCR    10,15
159       000A12   F8 71 D 1D0 6 01F                   ZAP    1D0(8,13),01F(2,6)     TS=01          DNM=2-329
          000A18   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          000A1E   F8 11 6 017 D 1D6                   ZAP    017(2,6),1D6(2,13)     DNM=2-207      TS=07
161       000A24   58 10 C 048                         L      1,048(0,12)            GN=012
          000A28   07 F1                               BCR    15,1
161       000A2A                             GN=014    EQU    *
          000A2A   F2 71 D 1DC 7 022                   PACK   1D0(8,13),022(2,7)     TS=01          DNM=1-320
          000A30   F9 11 D 1D6 C 068                   CP     1D6(2,13),068(2,12)    TS=07          LIT+3
          000A36   58 F0 C 054                         L      15,054(0,12)           GN=015
          000A3A   07 AF                               BCR    10,15
162       000A3C   F8 71 D 1DC 6 021                   ZAP    1D0(8,13),021(2,6)     TS=01          DNM=2-365
          000A42   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          000A48   F8 11 6 017 D 1D6                   ZAP    017(2,6),1D6(2,13)     DNM=2-207      TS=07
164       000A4E   58 10 C 048                         L      1,048(0,12)            GN=012
          000A52   07 F1                               BCR    15,1
164       000A54                             GN=015    EQU    *
          000A54   F8 71 D 1DC 6 023                   ZAP    1D0(8,13),023(2,6)     TS=01          DNM=2-402
          000A5A   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          000A60   F8 11 6 017 D 1D6                   ZAP    017(2,6),1D6(2,13)     DNM=2-207      TS=07
166       000A66                             GN=012    EQU    *
          000A66   F2 75 D 1D0 7 024                   PACK   1D0(8,13),024(6,7)     TS=01          DNM=1-352
          000A6C   FC 51 D 1D2 6 017                   MP     1D2(6,13),017(2,6)     TS=03          DNM=2-207
          000A72   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          000A78   F1 76 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(7,13)    TS=01          TS=01
          000A7E   F8 33 6 009 D 1D4                   ZAP    009(4,6),1D4(4,13)     DNM=2-95       TS=05
          000A84   94 0F 6 009                         NI     009(6),X'0F'           DNM=2-95
168       000A88   F2 75 D 1D0 7 024                   PACK   1D0(8,13),024(6,7)     TS=01          DNM=1-352
          000A8E   FC 51 D 1D2 6 019                   MP     1D2(6,13),019(2,6)     TS=03          DNM=2-233
          000A94   F1 75 D 1D0 D 1D0                   MVO    1D0(8,13),1D0(6,13)    TS=01          TS=01
          000A9A   F8 33 6 00D D 1D4                   ZAP    00D(4,6),1D4(4,13)     DNM=2-124      TS=05
170       000AA0   F8 73 D 1DC 6 00D                   ZAP    1D0(8,13),00D(4,6)     TS=01          DNM=2-124
          000AA6   FA 33 D 1D4 6 009                   AP     1D4(4,13),009(4,6)     TS=05          DNM=2-95
          000AAC   F2 75 D 1D8 7 024                   PACK   1D8(8,13),024(6,7)     TS=09          DNM=1-352
          000AB2   FB 33 D 1DC D 1D4                   SP     1DC(4,13),1D4(4,13)    TS=013         TS=05
          000AB8   F8 33 6 011 D 1DC                   ZAP    011(4,6),1DC(4,13)     DNM=2-150      TS=013
          000ABE   94 0F 6 011                         NI     011(6),X'0F'           DNM=2-150
172       000AC2   F8 73 D 1D8 6 011                   ZAP    1D8(8,13),011(4,6)     TS=09          DNM=2-150
```

```
                              CROSS-REFERENCE DICTIONARY

DATA NAMES                        DEFN     REFERENCE

PERSONNEL-MASTER-FILE             00012    00132  00132  00137  00141  00141
IM01-SOC-SEC-NUM                  00021    00175
IM01-NAME                         00022    00176
IM01-DEPARTMENT                   00023    00149  00197
IM01-NUMBER-DEPENDENTS            00025    00152  00155  00159  00161  00178
IM01-MONTHLY-PAY                  00026    00166  00168  00170  00177
PRINTER                           00014    00132  00132  00137  00185  00191  00198
OR01-PRINTER-RCD                  00030    00185  00185  00185  00191  00191  00191  00191  00198  00198  00198  00198
                                           00198  00198

WS-MASTER-FILE-EOF-SWITCH         00032    00134  00135  00144
WS-TOTAL-FAMILY-SIZE-SUM          00033    00184  00189  00193
WS-TOTAL-NET-PAY-SUM              00035    00183  00190  00193
WS-DEPARTMENT-HOLD                00037    00146  00146  00146  00149  00197
WS-FEDERAL-TAX-HOLD               00039    00166  00166  00170  00179
WS-FICA-TAX-HOLD                  00041    00168  00170  00181
WS-NET-PAY-HOLD                   00043    00170  00170  00173  00182  00183
WS-FAMILY-SIZE-HOLD               00045    00173  00184
WS-TAX-RATE-HOLD                  00047    00153  00156  00159  00162  00164  00166  00180
WS-FICA-TAX-RATE                  00049    00168
WS-0-DEPENDENTS-TAX-RATE          00051    00153
WS-1-4-DEPENDENTS-TAX-RATE        00054    00156
WS-5-8-DEPENDENTS-TAX-RATE        00057    00159
WS-9-12-DEPENDENTS-TAX-RATE       00060    00162
WS-13-99-DEPENDENTS-TAX-RATE      00063    00164
WS-SWITCH-ON                      00066    00135  00144
WS-SWITCH-OFF                     00068    00134
WS-COLUMN-HEADING                 00070    00198  00198
WS-DETAIL-LINE                    00089    00185  00185
WS-SOC-SEC-NUM                    00092    00175
WS-NAME                           00095    00176
WS-MONTHLY-PAY                    00098    00177
WS-NUMBER-DEPENDENTS              00101    00178
WS-PAY-AVERAGE                    00104    00173
WS-FEDERAL-TAX                    00107    00179
WS-TAX-RATE                       00110    00180
WS-FICA-TAX                       00113    00181
WS-NET-PAY                        00116    00182
WS-FOOTING-LINE                   00119    00191  00191
WS-TOTAL-FAMILY-SIZE              00125    00189
WS-TOTAL-NET-PAY                  00128    00190

PROCEDURE NAMES                   DEFN     REFERENCE

0010-READ-AND-PRINT               00140    00135
0010-EXIT                         00187    00135  00145
0020-TOTALS-RTN                   00188    00143  00150
0020-EXIT                         00195    00143  00150
```

```
28/08/78   PHASE   XFR-AD  LOCORE  HICORE  DSK-AD    ESD TYPE  LABEL     LOADED  REL-FR

      PHASE***  005000  005000  006213  2E 12 4  CSECT     DUMPOPE   005000  005000

                                                 CSECT     IJGFIS4Z  005E28  005E28
                                                 *  ENTRY  IJGFIZ4Z  005E28
                                                 *  ENTRY  IJGFIZZZ  005E28
                                                 *  ENTRY  IJGFIEZZ  005E28

                                                 CSECT     ILBDSAE0  006000  006000
                                                    ENTRY  ILBDSAE1  0060F0

                                                 CSECT     IJDFAPIZ  005010  005010
                                                 *  ENTRY  IJDFAZIZ  005010

                                                 CSECT     ILBDMNS0  0060C8  0060C8

* UNREFERENCED SYMBOLS                           WXTRN     STXITPSW
                                                 WXTRN     ILBDDBG2

002 UNRESOLVED ADDRESS CONSTANTS
```

```
          SSN                 NAME              MONTHLY PAY  NO OF DEPS PAY AVE   FED TAX    FICA TAX   NET-PAY



0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 005A8E - CONDITION CODE 0 - DATA EXCEPTION
0S00I JOB DUMPPE1  CANCELED
```

```
        DUMPPE1             28/C8/78                                                          22.42.59                    PAGE   1

GR 0-7  00005C82 C0005A66 000C54C0 CCCC5000   CC005298 50005CCE 000050E8 000053D8
GR 8-F  0C0C54C0 CC005C9E 00005000 0CCC5000   CCCC5768 00005548 80005C74 00005A54
COMREG  BG ACCR IS C00368

000C00  CCCCCCC0 CCC00000 0CCC4C6A 0C003439   CC000000 00000368 FF050000 00000000   ................  ................
000020  FFC50CC0 40004B96 FF15C007 CCCC5A94   5859C2C5 06D1F3C1 FF150130 50005DC2   .... ...........  $$BEDJ3A.......B
000040  000C1850 0CC00000 C0001528 CCCCC000   F7CE1800 026E29FC 00040000 0F002302   ...&............  7..............K
000C6C  CC040000 C0C376F4 0C04C000 CCC02392   CCCC0000 000011E4 00C40000 000386A4   ........4.......  ........U.......
000090  58F2F6F2 C6C2C8F6 C0CC0000 CCC00003   CC050C03 0003068C 06B00680 41880073   $262FBH6........  ................
0C0CA0  45700148 940FC393 41A0C658 45708180   18A84190 0178418C C1A047F0 C0DC0680   ......C....0....  .........A..0...
0000C0  068C0680 C6800680 C68CC68C 4188C017   4188CC50 45700148 41800178 9640A001   ................  ...&.......... ..
0000E0  9120A00C 471000EC 9260A001 95E2A002   4780848A 95C1A002 4780848A 9561A002   ...........-...S.  .....A......../..
000100  478C0110 9104AC00 478CC110 9203CC8F   9281A000 48A003C2 487AD604 49A003D6   ................  ........B..0....0
000120  41AAD648 C77894F9 7038C701 70587C58   9293AC0C 9680A001 440008EA 0788947F   ..0.....9..P.....  ................
00014C  AC014570 CC9A07F8 4280C0E9 58900288   9180A000 07175890 A0044880 02225000   ........B...Z....  ...............&.
000160  902C0218 90100298 D2C75C05 8CCC5680   AC0C07F7 45700150 D2078A32 8A3558E0   ..K......K.......  ...7...&K........
000180  02C49120 877A4780 018E9295 D6A8CC07   8A320648 18AA0008 8A32000C 43A10009   .D............0...  ..0..............
0001A0  42A00397 41AAD648 D2CCC589 03979540   05894740 01CC9560 05894750 01C69240   .......0.K.......  ... ...-......F.
0001C0  058547F0 01CCD200 0589C588 4590C2CA   47FCC662 5890A0C4 4220A000 9140A001   ...0...K.........  .0.............. ..
0C01E0  4710C8C6 D2070298 900845A0 09C241AA   C6040201 0016A000 98989010 82000298   ...FK..........B..  0.K..............
0C0200  4590C2CA 4400A004 5890A004 58C85C2C   91020021 4780021C 9200058A 989F0298   ................  ................
0C0220  92000020 928406A8 91E0877A 47800262   9120877A 47100254 9190886A 4710C254   .......0........  ................
000240  91038778 4770C178 9503C093 478CC178   47F00262 D2070298 8AC09890 8AC88200   ................  .0..K..........H..
0C0260  0298D207 02588A98 9890EAA0 82000299   96E8A000 41100030 47F08FBE 96030039   ..K..............  ...........0.......
000280  56010038 82000038 00615C40 61594044   FCCC0000 00000000 FF050007 40004884   ............/* /& .  0.............. ...
0002A0  00038FA8 50038D94 000370B8 0303AE70   CC034C8C 80001DAE 00001000 00002000   ................  ................
0C02CC  0C0C3000 00004000 3602C500 058905BA   C7899C68 05A09878 00505970 8A2447D0   ....... ..KN......  ..........&.......
0002E0  02EA8A70 CC0847F0 02EE8870 CCC81F78   18871C78 5E70058C 5080058C 913005BA   .........0.........  ..........&........
000300  4770034C 12774740 03401866 4A7005B0   5C600584 40600580 588005CC 18675E60   ........ . .......  .-.. -...........-
00032C  80205060 E0205880 05C81867 5E608020   5C6C8020 588005C4 5E708020 50708020   ..&-......H...-..  &-......D.....&...
000340  D200058A 05899868 05ACC7F9 18664360   05BA8860 00025886 05C05E70 801C5070   K.............9...-  ...-.............&.
000360  801C47F0 03400000 F2F861F0 F861F7F8   5C005000 00000000 00000000 00000000   ...0. ..28/08/78  &.&..............
0C0380  C4E4D4D7 C7C5F140 000107FF CC0C6213   CCCC6213 00000010 0003FFFF DAFD4E80   DUMPPE1 ..P......  ................+.
0C03A0  28A07EC0 006337C9 370C3991 39923A5C   3C703C74 3C7838F2 F8F0F8F7 F8F2F4F0   .........I.........*  ........280878240
0003C0  00003648 CC000014 18601996 1A3C1A4C   1A5C0020 378C0010 585BC206 00060007   ..........-.......  .*........$$BO....
0C03E0  01001804 36C40000 00000000 03681C80   CC0003F4 00000000 00003E7C 00000044   ................  ...4.............
0004CC  CC001A74 00000000 CCCCC0C0 CCCCCC00   CCCC0000 00000000 00000000 00000000   ................  ................
000420  00000000 00C04500 CC0C05A0 00003F38   CCCC0000 00000000 92380223 909F0298   ................  ................
0C0440  988E0288 47F0061C 47F0C646 4AA003C2   9588A000 47800468 9180A000 4710047A   ......0....0......B  ................
000460  5880AC04 9C08802C 988E0288 906905A0   459C04E0 986905A0 07F99601 A000988E   ................  ...........9......
000480  02889069 05A04590 04E09869 05A055FF   A00F0789 91A0877A 478004C2 9640877A   ................  .............B. ..
0C04A0  947F877A D2188AC9 02985000 8AE448E0   02220207 8AC0E000 94FD8AC1 988E0288   .....K..H..&..U..  ..K...........A.....
0004C0  07F95000 8A8C48E0 0222C207 8A58E000   58E002C4 94FD8A99 02188AA0 029807F9   .9&..........K......  ...D.....K........9
0C04E0  D2000589 C58A5878 0C5C5970 8A2447C0   04FA8A70 0C0847F0 04FE8870 00081F78   K..........&.......  ........0.........
0C0500  18871C78 5E7005BC 9180A000 4780051E   95FFA00F 4780051E 02000588 058A9130   ................  ...........K.......
0C0520  058A4770 C5729140 058AC719 508005BC   12774740 056C1866 4A700582 5D600584   ........ .....&...  ... ...........-..
000540  40600582 588005CC 18675E60 80245060   80245880 05C81867 5E608024 50608024    -...........-...&-  ......H...-...&-..
000560  588005C4 5E708024 5C7C8024 9240C58A   C7F99529 02230789 5080058C 18664360   ...D.....&..... ..  .9.......&.......-
000580  058A8860 00025886 05C05E70 801C5C70   801C9520 02230789 9240058A 07F90000   ...-............&.  ............ ...9..
0005A0  000021EE 90000178 0000C020 800005E4   CCCC0000 00000001 10101000 FE89A2AD   .................U  ..................
0005C0  00004258 CCC03F38 00004030 0C0C4129   CCCCC00C 00000050 909F0298 92200223   ............ .....  ........&.........
0005E0  459006C8 41900200 95000023 478008D2   95350023 4780060E 48600022 1A664570   ...H..............K  ............-......
C006C0  02C84866 7C0007F6 E2E5C3F1 F0F09564   C0234770 C0C09200 002307F1 41900080   .H......6SVC100..  ...............1....
```

```
0043AC  0C034C6C C0C30030 0003FFFF DAFD4E80   2C004E80 006337C9 37C03991 39923A5C    ...............*.  ..*....I........*
0C43CC  3C7C3C74 3C7830F2 F8F0F8F7 F8F2F4F0   C0003648 00000014 19201996 1A3C1A4C    .......280878240   ................
0043E0  1A5C0020 378CCC30 0C0CC0C0 CCCCCCC0   CCCC0G0C 360400C0 00000000 03680080    .*..............   ................
C044C0  C0004444 CCC0CC00 0C003E7C CCCCC044   CC001A74 0C00000C 00C00C00 0C000000    ................   ................
004420  0C0000CC CCC00000 CCCCC000 CCCCC000   CC000000 00000000 000005A0 00004030    ................   ............... .
00444C  CCCCCCCC CCC00000 0CC03E7C CCCCCC44   CCC01A74 0CC00CCC C0C00CCC 0C000000    ................   ................
00446C  0CCCCCCC CCC00000 0CC0C000 CCCC0000   CCC00000 00000000 000005A0 00004129    ................   ................
0044EC  CCG0C000 --SAME--                                                             ....
CC45C0  C3D9FCCC CC1C000F C3D7F000 C01C000F   C7C9F000 E01E000F C4C2FC13 0130130F    CR0......CP0......  PR0......DK0......
004520  C4D2F113 1131131F C4D2F213 2132132F   C4D2F313 3133133F C4D2F413 4134134F    DK1......DK2......  DK3......DK4......
0C4540  C4C2F514 0140140F C4D2F614 1141141F   C4D2F714 2142142F E3D7F018 0180180F    DK5.. ..DK6......   DK7......TP0......
0C456C  E3D7F118 1181181F E3D7F218 2182182F   E3C7F318 3183183F E3D7F418 4184184F    TP1......TP2......  TP3......TP4......
004580  E3C7F518 5185185F E3C4F018 0180180F   C5D5C44C 40000000 C5D5C44C 40004500    TP5......TD0......  END  ...END  ...
0045AC  CCCCCCCC --SAME--                                                             ....
004760  5B5BC2C4 E4D4D7C2 5B5BC2E3 D4C5C2C7   CCC00001 00000003 120+0478 5B5BC2E3    $$BDUMPB$$BTMEBG   .............$$BT
004780  F1F0F3F0 00C000C1 00000003 130103F1   5B5BC2E3 F1F0F5F0 00000001 00000003    1030............1   $$BT1050........
0C47AC  13020418 5B5BC2E3 F1F0F6FC 0CCCC001   CCC00003 1303030A 5B5BC2E3 F2F2F6F0    ....$$BT1060....   ........$$BT2260
0047C0  00000C01 CCC00003 13040497 5B5BC2E3   F2F7F4F0 00000001 00000004 00010499    .............$$BT   2740............
0047E0  5B5BC2E3 F2F8F4F8 CC000001 0CC0C004   0002049F 5B5BC2D3 C9E2E3E5 C0000001    $$BT2848........   ....$$BLISTV....
0048C0  0CCC0CC6 C6010208 5B5BC2C4 E4D40740   CCCC0001 00000006 06000305 5B5BC2C4    .........$$BDUMP   .............$$BD
004820  E4D4D7C2 CC000001 0000C006 C60403D8   5B5BC2C4 E4D4D7C4 C0000001 00000006    UMPB............0   $$BDUMPD........
004840  C7C104CC 5B5BC2C3 D3C4C1F1 CCC00001   CCC0C006 0801J0CE 5B5BC2C4 C9E3C1D5    ....$$BCLDA1....   .........$$BDITAN
004860  CC000001 GCC00004 C80200C6 5B5BC2C4   C9E3C6D7 00000001 00000006 080300E6    ............F$$BD   ITFP...........W
004880  5B5BC2E7 E2E6C3C8 CC000001 0CC00006   0804011A 5B5BC7E3 D7C9D240 C0000001    $$BXSWCH........   ....$$GT IK  ....
0048A0  CCCCCCC6 C9010523 5B5BC2C3 D9E9C1D7   CC0C0001 00000006 08040042 5B5BC2C4    .........$$BCRZAP   ..........  .$$BD
0048C0  E4D4D7C6 CC000001 00000006 0C01045C                                           UMPF...........*

L0LTYP  HEX LENGTH IS 0C00
--BG--
0C4FA0                    D5D640D5 C1D4C540   FF150007 C0005A94 00005C9E 00005000             NO NAME    ..........*...&.
004FC0  00005C00 C0005768 0C005548 80005C74   CC005A54 00005C82 00005A66 000054C0    ..&............*.   ......*.........
004FEC  CCCC5CCC C0C05288 5C005CCE 0C0050E8   C00053D8 000054C0 00000000 017582AE    ..&......&.*...&Y   ...Q............
005000  05FC070C 5CCEF0CA 47FCFC82 0CC050C0   CCCC5C0C 000107FF 00005000 0001C784    .0....0..00...&.   ..&...P...&...P.
005020  FFFFFF7C 0CC05000 0C0G4F88 0C006870   0A0407F1 00005010 00005010 00005FA8    ......&.........   ...1..&...&.....
00504C  C00C6FA8 CCC003E8 D2C2733E 7299F912   7341733E 4740703A FA107348 73CFF911    .........K.....9.   ..... ........9.
005060  73467348 47807014 D2117168 7276D202   7187728A D2087192 726DD202 71A37299    .........K.....K.   ....K.....K.....
005080  D283728A 58C0F0C6 58E0C000 58D0F0CA   55CCE030 4770F0A2 96100048 92FFE000    K......0F......0.   .......0........
0050A0  47F0F0AC 58CEF03A 90ECC00C 18509889F  F08A9110 D0480719 C7FF0700 00005C9E    .00...0.........   0..............*.
0050CC  00005CCC CCC05C00 00005768 C0005548   CC005800 00005C84 C3D6C2C6 F0F0F0F1    ..&...&.........   ......*.COBF0001
0050E0  C4E4D4D7 F0C7C540 F1D6C9D2 C9D5C7F0   F1000454 4CC7C540 C2C5C7C9 D5E24000    DUMPOPE 1ORKINGO   1....GE BEGINS .
005100  8CD9C520 CC150C12 0CC80C05 0CF0F1F0   40404040 40404040 40404040 4040E2E2    .RE...........010                 55
005120  D5404040 4C40404C 40404040 4C404040   D5C1D4C5 40404040 40404040 404040D4    N                    NAME            M
005140  D6D5E3C8 C3E84007 C1E84040 C5D64CD6   C640C4C5 D7E240D7 C1E840C1 E5C54040    ONTHLY PAY  NO 0   F DEPS PAY AVE
005160  40C6C5C4 4CE3C1E7 40404040 C6C9C3C1   40E3C1E7 404040D5 C5E360D7 C1E84040     FED TAX     FICA    TAX   NET-PAY
005180  40404040 40404040 4040E3C5 404040C6   4C404040 4040404C 404040C5 E34007C1              TE    F                ET PA
0051A0  E8404C40 4C404040 40404040 4040D7C1   40404040 40404040 40404040 40404040    Y                PA
0051CC  40404040 --SAME--
0051EC  40404040 40404040 40404040 40404040   406C4040 404040D9 40404040 40404040                         .    R
005200  404C4040 4C404040 40404040 40404040   40404040 404040C8 40404040 40404040                              H
005220  404C4040 4C404040 40404040 40404040   40404040 4040404C 404040C4 C5D7C1D9                                       DEPAR
005240  E3D4C5D5 E340E3D6 E3C1C340 C6C1D4C9   D3E840E2 C9E9C540 4040C4C5 D740C1D5    TMENT TOTAL FAMI   LY SIZE    DEP AN
005260  C44CE3D6 E3C1C340 D5C5E340 D7C1E840   4C40D3E8 40E2C9E9 C5404040 40404040    D TOTAL NET PAY      LY SIZE
005280  404C4C40 --SAME--
0052A0  000CCC50 CCC00000 CCC0C000 CCCCCCC0   CCCC0030 00000000 00008204 0C000106    ...&............   ................
```

```
        CUMPFE1              28/C8/78                                                                                              PAGE  12

0052C0  CC0C5320 3CC05340 0430CE28 2C0EE2E9   E2F0FCF6 40010000 00000000 00028020   ....... ......SY  S006 ...........
0052EC  CCCC0000 06005308 80C00000 CC000046   CCC0C000 00460000 02005904 C0C80013   ........Q.......  ..............H.
0053C0  2890CC4F CC46CC0C 010C60F0 5821CC58   CCC05308 0000005C 00005427 8E006000   ..........-0....  ...0...&......-.
005320  C70052F2 4C00C006 310052F4 400CCCC5   CP005328 C0C00C0C 06005390 00000050   ...2 ......4 ...  ...............&
005340  2890CC4F CCC0CC00 0C0C800C 0CC00135   CC005370 00005378 00005D10 00648BF1   ................  ...............1
005360  010054C1 CCCCGC00 C70C4120 ECCCCCCC   C1CC5431 20000084 06005418 00000050   ...A............  ...............&
0C5380  F6F3F9F1 F6F7F0F3 F9C8C6E3 E3400906   C2E84040 40404040 404040F0 F1F0F1D7   639167039HOTT RO  BY         0101P
C053A0  C6C3F0F9 F0F3F9F8 F0F04040 40404040   4C404040 40404040 40404040 40404040   FC09039800
0053C0  4C404C4C 40404040 40404040 40404040   CC1C58E0 F064C7FE F7F3F4F7 F6F4F0F2                    ....0...784764C2
0C53E0  F5C7D9E4 C5C4E840 C7C1D9E5 C5E840D1   C5404OF0 F1F0F1E2 C6C3F1F2 F0F5F6F8   5GRUNDY GARVEY J  R  0101SFC120568
00540C  F0F04C40 4C404040 4C40404C 40404040   4C404C40 40404040 40404040 40404040   DO
00542C  40404040 4C404040 E0004770 FC5A47F0   F14C4040 40404040 40404040 4040E2E2           ....0..0  1             SS
0C5440  C5404040 4C404040 40404040 4C4C4040   C5C1D4C5 40404040 40404040 404040D4   N                 NAME           M
005460  C6D5E3C8 C3E94C07 C1E84040 D5D64CD6   C84CC4C5 D7E240D7 C1E840C1 E5C54040   ONTHLY PAY  NO O  F DEPS PAY AVE
0C548C  40C6C5C4 4CE3C1E7 404C4040 C6C9C3C1   4CE3C1E7 40404005 C5E36007 C1E84040    FED TAX    FICA    TAX   NET-PAY
0C54A0  40404040 4C404C4C 40404040 404C4040   4C4C4C4C 40000000 47F0FC4C 0A320000                          ....00.....
0C54C0  0A32000C 47F0F01A C9D1C4C6 C3E9E9E6   F3F99180 10024710 F0240A07 90CEF060   .....00.IJDFCZZW  39.......0.....0-
0054E0  58E01C18 48C0102E 06C018D0 44C0F06C   58CEF060 D2001028 10170200 10171016   ..............0.  ..0-K......K.....
0055C0  GA0CC7FE 91901C02 4710F056 0A074200   1C280A0C C7FE0CCC 0CC00000 0C000000   ...........0....  ................
00552C  0CCCCCCC E2C05000 0000F01A 0A320000   C9D1C3C6 E9C9E9F0 030A0A00 91801002   ....K......0.....  IJCFZIZO........
005540  471CFC26 0AC750E0 3C6458E0 10209101   1C044780 F04C9140 10024710 F04658E0   ..0...&..........  ....0.. ....0...
005560  F06447FC F01A58E0 101C07FE E501FG60   ECC04770 F05A47F0 F04658E0 F06407FE   0..0P.......N.0-  ....0..00...0...
005580  615CCCCC CCC00000 0CF0F04C 0A32C000   3C0C0048 00000000 00000000 00005800   /*.......00......  ................
0C55A0  00CCC0CC CC0CC000 FC240A07 90CEFC60   58E01C18 48C01C2E 06C018D0 44C0F06C   ........0.....0-  ...............0.
0055CC  98CEF06C D2001028 1C17C200 10171016   C000004P 00000000 00000000 00000000   ..0-K.....K.....  ................
0055E0  CC0000C0 CCC00000 CCCCC00C CCCC0000   CCCC0C0C 0000000C 00000000 000000C0   ................  ................
0056CC  CCCC0000 --SAME--                                                            ....
CC56EC  CCCCCCCC CCC00C00 0C000000 CCCC0000   CCCC0030 00000000 00005000 00000000   ................  ..........&.....
0057CC  0C000000 C0005308 0C0054C0 000050E8   6C00528B 0000534B 0CC00000 0C56800F   ........0......&Y  ................
00572C  CCCCCCCC CCC00000 0CC0C000 CCCCCC00   CCC0000C 0000000C 0C000000 000058DC   ................  ..............$.
005740  0CCC5CC0 CCC05C82 CC00C470 0C0000E8   CCC05860 00005C4E 00005C92 000000E8   ..&...*.........V  ...-..*+..*....V
00576C  C00003C8 00C00398 000060C8 000058D4   C00058D6 000058DC 00005C4E 00005C7C   ...........-H...M  ..$0..$...*+..*.
00578C  C0CC5C82 CC005860 000C5872 CCCC5904   CCC0592E 0000591A 00005964 00005956   ..*....-........  ................
0057AC  0C0059AC C0005986 CC0059A4 000059D6   00005A66 00005AC0 00005A2A 00005A54   ...............0  ................
0057C0  C00C5C3C C00058DC 0C005C4E 00005C82   0F5C9C01 3C40205B 20202120 4B202040   ..*...$...*+..*.  .*... .$........
0057EC  202120FC 2C212000 CC0CC000 0CC00004   5858C206 07C5D540 5858C2C3 D3D6E2C5   ...0............  $$BOPEN $$BCLOSE
CC580C  581CC1C8 4810CC86 948F100C 4110CC88   58C0C1C8 184005FC 5C00F008 4500F00C   ..JH............  ..JH. .0&.0...0.
005820  0000529E CA024110 CC885800 D1CC1840   C7CC05F0 5000F008 4500F00C 00005348   ............J..   ...0&.0...0.....
005840  0A025020 D1C05880 D1C0C200 60006026   58000208 500001F4 5800C01C 50000208   ..&.J...J.K.-.-.  ..K.&.J4....&.K.
005860  5820C02G D5006000 60250782 5810CCC4   C7F15830 D1F45000 D2085800 D1C81840   ....N.-.-.......  .1..J4&.K...JH.
005880  4110C090 C70005F0 5000F0C8 4500F00C   CCC00000 0A025810 D1C84810 C08605F0   .......0&.0...0.  ........JH.....0
0058A0  91801000 4780F016 41101004 412010E0   D20F1000 20005800 D1CC1840 4110C090   ......0.........  K.......J.. ....
0058C0  C70C05FC 5C00F008 4500F00C 0CC00000   0A020A0E 581001C8 58F0C024 91201010   ...0&.0...0.....  ......JH.0......
0058E0  C71F1841 41F0C024 D2021041 F00158F0   101045E0 F0085020 D18C5870 D18C58F0   .....0..K...0..0  ....0.&.J...J..0
005900  C02807FF 58C0D20C 5C00C1F8 5800C02C   50C0020C 5810C00C 07F15800 D1F85000   ......K.&.J8....  &.K.......1..J8&.
0C5920  D20C02C0 60006025 5810C008 07F15820   C030954C 60070772 D5006308 60070772   K.K.-.-......1..  ... -...N.-.-...
005940  58000210 5C00D1FC 5800C034 5C00D210   5810C010 07F15800 D1FC5030 D2105810   ..K.&.J......&.K.  .....1..J.&.K...
00596C  C03807F1 5820C038 D50160C7 7018C782   5800020C 5000D20C 5800C03C 5000D20C   ...1....N.-.....  ..K.&.K......&.K.
0C5980  5810CC0C C7F1580C D20C500C D2CC5800   D2105000 D2045800 C0405000 D2105810   .....1..K.&.K...  K.&.K.... &.K...
0059A0  C01007F1 5800D204 5000C210 F2710100   7022F910 D1D6C068 58F0C044 077FF871   ...1..K.&.K.2.J.  ..9.JO...0.....8.
0C59CC  D1C06C18 F176D100 D1D0F811 6017C106   5810CC48 07F1F271 D1D07022 F910D106   J.-.1.J.J.8.-.JO  .....12.J...9.JO
0059E0  C06558FC CC4C07AF F871C100 601DF176   010CD1D0 F8116017 D1D65810 C04807F1   ...0....8.J.-.1.  J.J.8.-.JO.....1
005A00  F271D1C0 7022F910 D1D6C06A 58FCC050   G7AFF871 D1D0601F F1760100 D1D0F811   2.J...9.JO...0.&  ..8.J.-.1.J.J.8.
005A2C  6C17D1D6 5810C049 07F1F271 C1DC7022   F911D1D6 C06858F0 C05407AF F8710100   -.JO.....12.J...  9.JO...0....8.J.
```

```
005A40   6021F176 C1C0D10C F8116017 D1D65910    C04907F1 F8710100 6023F176 D100C103    -.1.J.J.8.-.JD..  ...18.J.-.1.J.J.
005A60   F8116017 C1C6F275 D1DC7024 FC51D102    6C17F176 D1D0D100 F1760100 D100F833    8.-.JO2.J......JK  -.1.J.J.1.J.J.8.
005A80   6CC9D1D4 94CF6C09 F275C1D0 7024FC51    C1C26019 F175C1C0 D1C0F833 600CD104    -.JM..-.2.J.....   JK-.1.J.J.8.-.JP
005AA0   F8730100 4000F833 D1D46009 F275D108    7024F833 D1DCD104 F8336011 D1DC940F    8.J.-...JM-.2.JQ   ....J.JM8.-.J...
005AC0   4011F873 C1D86011 FC51C1CA 6015F873    C1C9C1DA D209D1E0 C0604110 D1E6DF09    -.8.JO-...J.-.8.   JQJ.K.J......JW..
005AE0   C1E0C1CC C6109258 1C0CC2C7 6CEFD1E2    C2C96C93 70000211 60C17009 F275C1D8    J.J.....$..K.-.JS  K.-...K.-A..2.JC
005B00   7024C209 C150C06D 4110C1E6 CFC9C150    C1DCC610 92581060 D2C76009 D1E2F271    ..K.J......JW..J.  J....$..K.-RJS2.
005B20   D1C87C22 C2C3D1FD CC77C503 D1E0C1CE    C2D16CE7 C1E20209 D1C0C06D 4110C1E6    JQ..K.J.......J.J. K.-XJSK.J......JW
005B40   CF39C15C 6CC9D610 92581000 C2076CF9    C1E2F871 D1D96017 C7C5D1D9 D1D9940F    ..J.-....$..K.-9   JS8.JQ-.P.JQJQ..
005B60   D1DE0203 D1E0CC78 CE03C1E0 C1DEC201    6106C1E2 D209D1E0 C0604110 D1E6DF09    J.K.J......J.J.K.  /.JSK.J......JW..
005B80   C1E06C0C C6109259 1C0CC207 610FC1E2    C2C9C1E0 C0604110 D1E60F09 D1E06011    J.-....$..K./.JS   K.J......JW..J.-.
005BA0   06109258 1C0C02C7 6114C1E2 FA3360C3    6C11FA11 60016015 02949000 60A85810    ...$..K./.JS..-.   -...-.-.K...-...
005BC0   D1CC92FC 8C0058F0 101045E0 FCCC5020    C1CC5890 D1C05610 D20907F1 D203C1E0    J..0...0.....J.&.  J...J...K..1K.J.
005BE0   C0770E03 C1506001 C2026172 D1E1F873    D1096003 D7030108 D103940F D10CD209    ....J.-.K./.J.8.   JQ-.P.JQJQ..J.K.
005C00   C1E0CC60 4110D1F6 CFC9C1E0 C10C0610    92581000 D207618A D1E20293 80006130    J.....JW..J.J...   .$..K./.JSK.../.
005C20   92408CE4 5810D1CC 926C8000 58FC1010    45E0FC0C 502001C0 580001C0 D2016001    . ....J..-...0..   ..0.&.J...J.K.-.
005C40   C07FD203 60C3C081 5810C20C 07F10201    6CC77C19 D2793000 60299240 807AD209    ..K.-.....K..1K.   -...K...-.. ..K.
005C60   8C7A8C7A 5810D1CC 92F18000 58FC1010    45E0F00C 502001C0 5880D1C0 5810C210    ......J..1...0..   ..0.&.J...J...K.
005C80   C7F10A0E 50005008 5050D004 582CC000    55C02000 077992FF 20009610 D04950E0    .1..&.&.&&......   ...............&.
005CA0   D05405FC 9120D048 47E0F016 58C08C48    582D8050 58E00054 07FE9620 D0484160    ...0.......0......  ...&.............-
005CC0   00044110 C0044170 CC68C670 055C5840    1CCC1E48 50401000 87165000 4180D18C    ...............&.  ....& ....&...J.
005CE0   417001CF C5105800 80001E08 5C008000    E7861000 D2080208 C05C5860 D1C45870    ..J............&...  ....K.K..*.-JD..
005D00   D18C5980 C1C058E0 C054C7FE 7CCC96F0    CA320C30 47F0F084 0A320000 47F0F01A    J...J..............0  .....00........00.
005D20   C9D1C4C6 C1D7C9E9 F3F9918C 10024710    F0240A07 90CEF0EC 58E01018 06E0D200    IJDFAPIZ39......   0......0.........K.
005D40   1017E000 18CC180C 43C01017 41E00010    43DEF0F7 19DC4780 F05046E0 F0400A32    ................   ..07....0&..0 ..
005D60   43CEF107 43C01016 19CD4770 F0769180    10164780 F0769208 10280A00 91801002    ..1..........0...  ....0............
005D80   4710FC76 0AC742D0 10284200 10160A00    91901002 4710F08A 0A07D201 10261003    ..0..............  ......0...K.....
005DA0   92011028 58E01C28 D2021029 10195CE0    1C18C6E0 98CDF0EC 44001022 58E0F0F4    .........K......&.  .......0........04
005DC0   0A0007FE 91801CC2 4710F08E 0AC79101    10154780 F00494FE 10159102 1026C78E    ...........0......  ....0M...........
005DE0   47F0F0CA 91011027 C78E1200 4780FCE4    18F0C7FF 92881028 0A0007FE 00005768    .00.............0U  .0...............
005E00   00005548 80C05C74 F2C2C1F9 F8F7F6F5    F4F3C3F1 4E60F040 930803C8 C38883A8    ......*.2BA98765   43C1+-0 ..L.C...
005E20   A398E388 C3181308 47F0F014 47F0F252    47F0F022 47F0F014 47F0F014 0A32C9D1    ..T.......00..02.  .C0..00..00...IJ
005E40   C7C6C9C5 E6E9F3F9 000890AE F26C9180    10494710 F04A58A0 102C50A0 10584AA0    GFIEWZ39.....2...  ....0......&.....
005E60   10485CA0 10609140 10274710 F1CC47F0    F0A0988D 105886BC FC885090 10589110    ..&..-. ....1..0   0........0.&.....
005E80   10154780 FC7E180C 41A0C1C0 18CA47C0    FC76D2FF D0008000 1AD886BA F06441CC    ....0............  0.K...........0...
005EA0   00FF44CC F25A98AE F26C4400 105407FE    91C81049 4780F098 94F71049 47F0F1CC    ....2...2........  .......0..7...01.
005EC0   91081C64 4710F0A2 0A0C9180 10024710    F0AC0A07 91011004 4780F098 47F0F236    .......0...........  0...........0..02.
005EE0   91C91003 4780F0C4 91801064 4780FCD0    58E01C64 47F0F0E4 91921003 4780F110    ......00........0.  .....00U........1.
005F00   91801064 4780F14C 58E01064 98ADF26C    90F1F100 58101080 070005EE 98F1E00C    .......1........2.  .11.............1..
005F20   90ACF26C 47F0F154 00000000 0CCC0000    CC0C0C00 0C000000 91401005 4780F158    ..2..01...........  ........... ....1.
005F40   48810CCC 4980F260 4780F13C 4880105E    4980F260 4720F124 4740F13C 96401049    .......2-..1......  ..2-..1.. 1.. ..
005F60   47FCF158 91041064 478CF0D8 58E01C50    47FCFCE4 91201064 4780F158 96101049    .01.........00....&  .00U........1.....
005F80   598C1C50 4138000C 50801058 914C1049    4710F178 4A80104A 50801060 47F0F18A    .........&..... ..  ..1.......&..-.01.
005FA0   48C0104A 49C1C0C0 1AC850C0 1C6C5740    104948E0 103489E0 000843E0 F2634180    ......A.....&..-.  ...............2...
005FC0   000518CD 43C81C38 43D81043 19CD4770    F18642E8 103888E0 00084650 F19C41CC    ..................  1..............1...
005FE0   000142C8 1038D503 103C1036 47D0F202    96081049 9001F280 98A0F26C 4100F284    ........N.........2.  .......2...2...2.
006000   4110F264 CAC29801 F2809120 10104710    FC7E9140 1027478C F1FA949F 102747F0    ..2.......2........  0.. ....1.......0
006020   F0A091C8 1C494710 F21A9108 10644780    F21A5880 10800202 10811020 5080102C    0.........2........  2......K.......&...
006040   0A009110 10494780 F22A9710 1C4947F0    FC289680 1049988D 105847F0 F0569180    ..........2........0  0..............00...
006060   10264710 F1CC9120 10274710 F1CC94FE    1C495640 102747F0 F1CCD203 10581060    .....1.........1...  ... ...01.K.....-
006080   C7FED20C CCC08000 0000C001 5858C206    C7C5D540 00005000 00005000 00005768    ..K............$$80  PEN ..&....&.....
0060A0   C0005548 8C0058F6 0C0C0000 CCCCCCC0    FFCC0C00 00000000 00000000 00000000    ........6..........  ..................
0060C0   CC0CCC0C CCC0C7FE FF081049 4780FC99    C5F047F0 F019C9D3 C2C4E2C1 C5F0F3F1    .................0.  .0.00.ILBDSAE031
0060E0   F4F3F0F3 61F0F761 F7F41800 47F0F022    41C0C001 05F050EC FCE89005 F0F24144    4303/07/74...00.   .....0&.0...02..
```

```
         DUMPFE1              29/08/78                                                                                  PAGE  14

006100   00005110 40144780 FC305108 4015471C    F030912C 40024780 F02C4100 001047F0      .... ...0... ...  0... ...0......0
006120   F0305811 00044154 000C4850 F11A5E55    CCCC4155 00001255 4780F0AC 18350501      0...........&1...  ............0...N.
006140   F0E43C00 4780FC5A 41330002 47F0F049    5C50F0EA 05033006 F0E04750 F09E4403      0U....0.......00.  &&0.N...0...0...
006160   00064403 C0CA1852 92FC5000 D2C65001    5CCC1200 4780F09A 4900F0E6 4780F092      ..........0&.K.&.  &......0...0M..0.
006180   56015002 47F0F09E 56015001 47F0F095    56015000 59E0F0EE 9805F0F2 58F0F0EA      ..&..00...&..00.  ..&...0...02.00.
0061A0   07FF95C0 C0184940 F11C1604 4110F13A    C8C2910E 00494780 F0005810 F11258F0      ....... 1.....1.  ..........0...1..0
0061C0   F1160701 10C21002 4111003C 50010000    C7FF1900 0A060000 000005F0 00011035      1.P...........&J..  ...............0....
0061E0   43C91043 19CD4770 F18642E8 1C38E8E5    CCC84690 F19C41CC 000142C8 10380503      .........1........  ....1...........N.
006200   5E58C2C3 D6C2C5D9 00000000 00000000    00040008 00000000 00000000 00000000      *8BCOBER.........  ..................
006220   00000000 --SAME--                                                                .-..
0107E0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000      ..................  ..................
```

287

DEBUG

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 3

PROGRAM NAME: DUMP-PE

```
CBL LIB
00001          BASIS C32DPP
00002   000010 IDENTIFICATION DIVISION.
00003   000020 PROGRAM-ID. DUMP-PF.
00004   000030 AUTHOR. LT HUDGIN.
00005   000040 REMARKS. BE DESIGNED FOR USING ALC TO DEBUG A COBOL PROGRAM.
00006   000050 ENVIRONMENT DIVISION.
00007   000060 CONFIGURATION SECTION.
00008   000070 SPECIAL-NAMES.
00009   000080     C01 IS CHAN1.
00010   000090 INPUT-OUTPUT SECTION.
00011   000100 FILE-CONTROL.
00012   000110     SELECT PERSONNEL-MASTER-FILE
00013   000120       ASSIGN TO SYS006-UT-2314-S.
00014   000130     SELECT PRINTER
00015   000140       ASSIGN TO SYS005-UR-1403-S.
00016   000150 DATA DIVISION.
00017   000160 FILE SECTION.
00018   000170 FD  PERSONNEL-MASTER-FILE
00019   000180     LABEL RECORDS STANDARD.
00020   000190 01  IM01-PERSONNEL-MASTER-RCD.
00021   000200     05  IM01-SOC-SEC-NUM                 PIC X(9).
00022   000210     05  IM01-NAME                        PIC X(18).
00023   000220     05  IM01-DEPARTMENT                  PIC X(2).
00024   000230     05  FILLER                           PIC X(5).
00025   000240     05  IM01-NUMBER-DEPENDENTS           PIC 9(2).
00026   000250     05  IM01-MONTHLY-PAY                 PIC 9(4)V9(2).
00027   000260     05  FILLER                           PIC X(38).
00028   000270 FD  PRINTER
00029   000280     LABEL RECORDS OMITTED.
00030   000290 01  OR01-PRINTER-RCD                     PIC X(133).
00031   000300 WORKING-STORAGE SECTION.
00032   000310 77  WS-MASTER-FILE-EOF-SWITCH            PIC X.
00033   000320 77  WS-TOTAL-FAMILY-SIZE-SUM             PIC S9(3)
00034   000330                                  USAGE IS COMP-3.
00035   000340 77  WS-TOTAL-NET-PAY-SUM                 PIC S9(5)V9(2)
00036   000350                                  USAGE IS COMP-3.
00037   000360 77  WS-DEPARTMENT-HOLD                   PIC X(2)
00038   000370                                  VALUE IS SPACES.
00039   000380 77  WS-FEDERAL-TAX-HOLD                  PIC S9(4)V9(2)
00040   000390                                  USAGE IS COMP-3.
00041   000400 77  WS-FICA-TAX-HOLD                     PIC S9(4)V9(2)
00042   000410                                  USAGE IS COMP-3.
00043   000420 77  WS-NET-PAY-HOLD                      PIC S9(4)V9(2)
00044   000430                                  USAGE IS COMP-3.
00045   000440 77  WS-FAMILY-SIZE-HOLD                  PIC S9(3)
00046   000450                                  USAGE IS COMP-3.
00047   000460 77  WS-TAX-RATE-HOLD                     PIC S9V9(2)
00048   000470                                  USAGE IS COMP-3.
00049   000480 77  WS-FICA-TAX-RATE                     PIC SV9(3)
00050 1                                         VALUE IS .058
00051   000490                                  USAGE IS COMP-3.
00052   000500 77  WS-0-DEPENDENTS-TAX-RATE             PIC SV9(3)
```

```
00053   000510                                     USAGE IS COMP-3
00054   000520                                     VALUE IS +.2.
00055   000530 77  WS-1-4-DEPENDENTS-TAX-RATE             PIC SV9(3)
00056   000540                                     USAGE IS COMP-3
00057   000550                                     VALUE IS +.15.
00058   000560 77  WS-5-8-DEPENDENTS-TAX-RATE             PIC SV9(3)
00059   000570                                     USAGE IS COMP-3
00060   000580                                     VALUE IS +.12.
00061   000590 77  WS-9-12-DEPENDENTS-TAX-RATE            PIC SV9(3)
00062   000600                                     USAGE IS COMP-3
00063   000610                                     VALUE IS +.08.
00064   000620 77  WS-13-99-DEPENDENTS-TAX-RATE           PIC SV9(3)
00065   000630                                     USAGE IS COMP-3
00066   000640                                     VALUE IS +.05.
00067   000650 77  WS-SWITCH-ON                           PIC X
00068   000660                                     VALUE IS '0'.
00069   000670 77  WS-SWITCH-OFF                          PIC X
00070   000680                                     VALUE IS '1'.
00071   000690 01  WS-COLUMN-HEADING.
00072   000700     05  FILLER                             PIC X(14)
00073   000710                                     VALUE IS SPACES.
00074   000720     05  FILLER                             PIC X(18)
00075   000730                                     VALUE IS 'SSN'.
00076   000740     05  FILLER                             PIC X(15)
00077   000750                                     VALUE IS 'NAME'.
00078   000760     05  FILLER                             PIC X(13)
00079   000770                                     VALUE IS 'MONTHLY PAY'.
00080   000780     05  FILLER                             PIC X(11)
00081   000790                                     VALUE IS 'NO OF DEPS'.
00082   000800     05  FILLER                             PIC X(10)
00083   000810                                     VALUE IS 'PAY AVE'.
00084   000820     05  FILLER                             PIC X(11)
00085   000830                                     VALUE IS 'FED TAX'.
00086   000840     05  FILLER                             PIC X(11)
00087   000850                                     VALUE IS 'FICA TAX'.
00088   000860     05  FILLER                             PIC X(19)
00089   000870                                     VALUE IS 'NET-PAY'.
00090   000880 01  WS-DETAIL-LINE.
00091   000890     05  FILLER                             PIC X(11)
00092   000900                                     VALUE SPACES.
00093   000910     05  WS-SOC-SEC-NUM                     PIC X(9).
00094   000920     05  FILLER                             PIC X(5)
00095   000930                                     VALUE IS SPACES.
00096   000940     05  WS-NAME                            PIC X(18).
00097   000950     05  FILLER                             PIC X(6)
00098   000960                                     VALUE SPACES.
00099   000970     05  WS-MONTHLY-PAY                     PIC $$$$9.99.
00100   000980     05  FILLER                             PIC X(6)
00101   000990                                     VALUE SPACES.
00102   001000     05  WS-NUMBER-DEPENDENTS               PIC Z9.
00103   001010     05  FILLER                             PIC X(8)
00104   001020                                     VALUE IS SPACES.
00105   001030     05  WS-PAY-AVERAGE                     PIC $$$$9.99.
```

```
001040     05  FILLER                          PIC X(2)
001050                                 VALUE IS SPACES.
001060     05  WS-FEDERAL-TAX                  PIC $$$$9.99.
001070     05  FILLER                          PIC X(5)
001080                                 VALUE IS SPACES.
001090     05  WS-TAX-RATE                     PIC VZ9.
001100     05  FILLER                          PIC X(7)
001110                                 VALUE IS ' %'.
001120     05  WS-FICA-TAX                     PIC $$$$9.99.
001130     05  FILLER                          PIC X(3)
001140                                 VALUE IS SPACES.
001150     05  WS-NET-PAY                      PIC $$$$9.99.
001160     05  FILLER                          PIC X(11)
001170                                 VALUE IS SPACES.
001180 01  WS-FOOTING-LINE.
001190     05  FILLER                          PIC X(35)
001200                                 VALUE IS SPACES.
001210     05  FILLER                          PIC X(31)
001220                                 VALUE IS
001230         'DEPARTMENT TOTAL FAMILY SIZE'.
001240     05  WS-TOTAL-FAMILY-SIZE            PIC ZZ9.
001250     05  FILLER                          PIC X(21)
001260                                 VALUE IS ' AND TOTAL NET PAY'.
001270     05  WS-TOTAL-NET-PAY                PIC $$$$9.99.
001280     05  FILLER                          PIC X(34)
001290                                 VALUE IS SPACES.
001300 PROCEDURE DIVISION.
001310     OPEN INPUT PERSONNEL-MASTER-FILE
001320          OUTPUT PRINTER.
001330     MOVE WS-SWITCH-OFF TO WS-MASTER-FILE-EOF-SWITCH.
001340     PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT
001350       UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.
001360     CLOSE PERSONNEL-MASTER-FILE
001370           PRINTER.
001380     STOP RUN.
001390 0010-READ-AND-PRINT.
001400     READ PERSONNEL-MASTER-FILE
001410       AT END
001420         PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
001430         MOVE WS-SWITCH-ON TO WS-MASTER-FILE-EOF-SWITCH
001440         GO TO 0010-EXIT.
001450     IF WS-DEPARTMENT-HOLD EQUAL TO SPACES
001460       PERFORM 0030-HEADING-RTN THRU 0030-EXIT
001470     ELSE
001480       IF WS-DEPARTMENT-HOLD NOT EQUAL TO IM01-DEPARTMENT
001490         PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
001500         PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
001510     IF IM01-NUMBER-DEPENDENTS EQUAL TO ZEROS
001520       MOVE WS-0-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
001530     ELSE
001540       IF IM01-NUMBER-DEPENDENTS LESS THAN 5
001550         MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
001560       ELSE
```

```
001570         IF IM01-NUMBER-DEPENDENTS LESS THAN 9
001580           MOVE WS-5-8-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
001590         ELSE
001600           IF IM01-NUMBER-DEPENDENTS LESS THAN 13
001610             MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
001620           ELSE
001630             MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
001640              WS-TAX-RATE-HOLD.
001650     MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
001660       GIVING WS-FEDERAL-TAX-HOLD.
001670     MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
001680       GIVING WS-FICA-TAX-HOLD.
001690     SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
001700       IM01-MONTHLY-PAY
001710       GIVING WS-NET-PAY-HOLD
001720     DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
001730       GIVING WS-PAY-AVERAGE.
001740     MOVE IM01-SOC-SEC-NUM TO WS-SOC-SEC-NUM.
001750     MOVE IM01-NAME TO WS-NAME.
001760     MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
001770     MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
001780     MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
001790     MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
001800     MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
001810     MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
001820     ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
001830     ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
001840     WRITE OR01-PRINTER-REC FROM WS-DETAIL-LINE
001850       AFTER ADVANCING 2 LINES.
001860 C010-EXIT. EXIT.
001870 C020-TOTALS-RTN.
001880     MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
001890     MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
001900     WRITE OR01-PRINTER-REC FROM WS-FOOTING-LINE
001910       AFTER ADVANCING 3 LINES.
001920     MOVE ZEROS TO WS-TOTAL-FAMILY-SIZE-SUM
001930       WS-TOTAL-NET-PAY-SUM.
001940 C020-EXIT. EXIT.
001950 C030-HEADING-RTN.
001960     MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
001970     WRITE OR01-PRINTER-REC FROM WS-COLUMN-HEADING
001980       AFTER ADVANCING CHAN1.
001990 C030-EXIT. EXIT.
```

```
INTRNL NAME    LVL SOURCE NAME                          BASE    DISPL  INTRNL NAME   DEFINITION   USAGE        R O Q M

DNM=1-151      FD  PERSONNEL-MASTER-FILE                DTF=01         DNM=1-151                  DTFSD              F
DNM=1-196      01  IM01-PERSONNEL-MASTER-RCD            BL=1    000    DNM=1-196     DS 0CL80     GROUP
DNM=1-234      02  IM01-SOC-SEC-NUM                     BL=1    000    DNM=1-234     DS 9C        DISP
DNM=1-260      02  IM01-NAME                            BL=1    009    DNM=1-260     DS 18C       DISP
DNM=1-279      02  IM01-DEPARTMENT                      BL=1    01B    DNM=1-279     DS 2C        DISP
DNM=1-304      02  FILLER                               BL=1    01D    DNM=1-304     DS 5C        DISP
DNM=1-320      02  IM01-NUMBER-DEPENDENTS               BL=1    022    DNM=1-320     DS 2C        DISP-NM
DNM=1-352      02  IM01-MONTHLY-PAY                     BL=1    024    DNM=1-352     DS 6C        DISP-NM
DNM=1-378      02  FILLER                               BL=1    02A    DNM=1-378     DS 38C       DISP
DNM=1-397      FD  PRINTER                              DTF=02         DNM=1-397                  DTFPR              F
DNM=1-428      01  OR01-PRINTER-RCD                     BL=2    000    DNM=1-428     DS 133C      DISP
DNM=1-454      77  WS-MASTER-FILE-EOF-SWITCH            BL=3    000    DNM=1-454     DS 1C        DISP
DNM=2-000      77  WS-TOTAL-FAMILY-SIZE-SUM             BL=3    001    DNM=2-000     DS 2P        COMP-3
DNM=2-034      77  WS-TOTAL-NET-PAY-SUM                 BL=3    003    DNM=2-034     DS 4P        COMP-3
DNM=2-067      77  WS-DEPARTMENT-HOLD                   BL=3    007    DNM=2-067     DS 2C        DISP
DNM=2-095      77  WS-FEDERAL-TAX-HOLD                  BL=3    009    DNM=2-095     DS 4P        COMP-3
DNM=2-124      77  WS-FICA-TAX-HOLD                     BL=3    00D    DNM=2-124     DS 4P        COMP-3
DNM=2-150      77  WS-NET-PAY-HOLD                      BL=3    011    DNM=2-150     DS 4P        COMP-3
DNM=2-178      77  WS-FAMILY-SIZE-HOLD                  BL=3    015    DNM=2-178     DS 2P        COMP-3
DNM=2-207      77  WS-TAX-RATE-HOLD                     BL=3    017    DNM=2-207     DS 2P        COMP-3
DNM=2-233      77  WS-FICA-TAX-RATE                     BL=3    019    DNM=2-233     DS 2P        COMP-3
DNM=2-259      77  WS-0-DEPENDENTS-TAX-RATE             BL=3    01B    DNM=2-259     DS 2P        COMP-3
DNM=2-293      77  WS-1-4-DEPENDENTS-TAX-RATE           BL=3    01D    DNM=2-293     DS 2P        COMP-3
DNM=2-329      77  WS-5-8-DEPENDENTS-TAX-RATE           BL=3    01F    DNM=2-329     DS 2P        COMP-3
DNM=2-365      77  WS-9-12-DEPENDENTS-TAX-RATE          BL=3    021    DNM=2-365     DS 2P        COMP-3
DNM=2-402      77  WS-13-99-DEPENDENTS-TAX-RATE         BL=3    023    DNM=2-402     DS 2P        COMP-3
DNM=2-440      77  WS-SWITCH-ON                         BL=3    025    DNM=2-440     DS 1C        DISP
DNM=2-462      77  WS-SWITCH-OFF                        BL=3    026    DNM=2-462     DS 1C        DISP
DNM=3-000      01  WS-COLUMN-HEADING                    BL=3    028    DNM=3-000     DS 0CL122    GROUP
DNM=3-030      02  FILLER                               BL=3    028    DNM=3-030     DS 14C       DISP
DNM=3-049      02  FILLER                               BL=3    036    DNM=3-049     DS 18C       DISP
DNM=3-068      02  FILLER                               BL=3    048    DNM=3-068     DS 15C       DISP
DNM=3-087      02  FILLER                               BL=3    057    DNM=3-087     DS 13C       DISP
DNM=3-106      02  FILLER                               BL=3    064    DNM=3-106     DS 11C       DISP
DNM=3-125      02  FILLER                               BL=3    06F    DNM=3-125     DS 10C       DISP
DNM=3-144      02  FILLER                               BL=3    079    DNM=3-144     DS 11C       DISP
DNM=3-163      02  FILLER                               BL=3    084    DNM=3-163     DS 11C       DISP
DNM=3-182      02  FILLER                               BL=3    08F    DNM=3-182     DS 19C       DISP
DNM=3-201      01  WS-DETAIL-LINE                       BL=3    0A8    DNM=3-201     DS 0CL133    GROUP
DNM=3-231      02  FILLER                               BL=3    0A8    DNM=3-231     DS 11C       DISP
DNM=3-250      02  WS-SOC-SEC-NUM                       BL=3    0B3    DNM=3-250     DS 9C        DISP
DNM=3-274      02  FILLER                               BL=3    0BC    DNM=3-274     DS 5C        DISP
DNM=3-293      02  WS-NAME                              BL=3    0C1    DNM=3-293     DS 18C       DISP
DNM=3-310      02  FILLER                               BL=3    0D3    DNM=3-310     DS 6C        DISP
DNM=3-329      02  WS-MONTHLY-PAY                       BL=3    0D9    DNM=3-329     DS 8C        NM-EDIT
DNM=3-364      02  FILLER                               BL=3    0E1    DNM=3-364     DS 6C        DISP
DNM=3-383      02  WS-NUMBER-DEPENDENTS                 BL=3    0E7    DNM=3-383     DS 2C        NM-EDIT
DNM=3-420      02  FILLER                               BL=3    0E9    DNM=3-420     DS 6C        DISP
DNM=3-439      02  WS-PAY-AVERAGE                       BL=3    0EF    DNM=3-439     DS 8C        NM-EDIT
DNM=3-474      02  FILLER                               BL=3    0F7    DNM=3-474     DS 2C        DISP
```

```
              MEMORY MAP

                   TGT                           00548

               SAVE AREA                         00548
               SWITCH                            00590
               TALLY                             00594
               SORT SAVE                         00598
               ENTRY-SAVE                        0059C
               SORT CORE SIZE                    005A0
               ASTC-REELS                        005A4
               SORT RET                          005A6
               WORKING CELLS                     005A8
               SORT FILE SIZE                    006D8
               SORT MODE SIZE                    006DC
               PGT-VN TBL                        006E0
               TGT-VN TBL                        006E4
               SORTAB ADDRESS                    006E8
               LENGTH OF VN TBL                  006EC
               LNGTH OF SORTAB                   006EE
               PGM ID                            006F0
               A(INIT1)                          006F8
               UPSI SWITCHES                     006FC
               OVERFLOW CELLS                    00704
               BL CELLS                          00704
               DTFADR CELLS                      00710
               TEMP STORAGE                      00718
               TEMP STORAGE-2                    00728
               TEMP STORAGE-3                    00738
               TEMP STORAGE-4                    0073A
               BLL CELLS                         00738
               VLC CELLS                         0073C
               SBL CELLS                         0073C
               INDEX CELLS                       0073C
               SUBADR CELLS                      0073C
               ONCTL CELLS                       0073C
               PFMCTL CELLS                      0073C
               PFMSAV CELLS                      0073C
               VN CELLS                          00750
               SAVE AREA =2                      0075C
               XSASW CELLS                       0075C
               XSA CELLS                         0075C
               PARAM CELLS                       0075C
               RPTSAV AREA                       0075C
               CHECKPT CTR                       0075C
               IOPTR CELLS                       0075C

LITERAL POOL (HEX)

007D0 (LIT+0)     0F5C9C01  3C40205A  20202120  4B202040  202120FC  20212000
007E8 (LIT+24)    0C0000C0  0C000004  5B5BC2C6  C7C5C540  5B5BC2C3  D3D6E2C5
```

```
            PGT                         00768

          OVERFLOW CELLS                00768
          VIRTUAL CELLS                 00768
          PROCEDURE NAME CELLS          0076C
          GENERATED NAME CELLS          00784
          SUBDTF ADDRESS CELLS          007C4
          VNI CELLS                     007C4
          LITERALS                      007C0
          DISPLAY LITERALS              008C0

REGISTER ASSIGNMENT

 REG 6   BL =3
 REG 7   BL =1
 REG 8   BL =2

133             000800                        START    EQU    *
                000800  58 10 D 1C8                    L      1,1C8(0,13)         DTF=1
                000804  48 10 C 086                    SH     1,086(0,12)         LIT+30
                000808  94 BF 1 000                    NI     000(1),X'BF'
                00080C  41 10 C 088                    LA     1,088(0,12)         LIT+32
                000810  58 00 D 1C8                    L      0,1C8(0,13)         DTF=1
                000814  18 40                          LR     4,0
                000816  05 F0                          BALR   15,0
                000818  50 00 F 008                    ST     0,008(0,15)
                00081C  45 00 F 00C                    BAL    0,00C(0,15)
                000820  00000000                       DC     X'00000000'
                000824  0A 02                          SVC    2
                000826  41 10 C 088                    LA     1,088(0,12)         LIT+32
                00082A  58 00 D 1CC                    L      0,1CC(0,13)         DTF=2
                00082E  18 40                          LR     4,0
                000830  07 00                          BCR    0,0
                000832  05 F0                          BALR   15,0
                000834  50 00 F 008                    ST     0,008(0,15)
                000838  45 00 F 00C                    BAL    0,00C(0,15)
                00083C  00000000                       DC     X'00000000'
                000840  0A 02                          SVC    2
                000842  50 20 D 1C0                    ST     2,1C0(0,13)         BL =2
                000846  58 80 D 1C0                    L      8,1C0(0,13)         BL =2
135             00084A  D2 00 6 000 6 026              MVC    000(1,6),026(6)     DNM=1-454    DNM=2-462
136             000850  58 00 D 208                    L      0,208(0,13)         VN=01
                000854  50 00 D 1F4                    ST     0,1F4(0,13)         PSV=1
                000858  58 00 C 01C                    L      0,01C(0,12)         GN=01
                00085C  50 00 D 208                    ST     0,208(0,13)         VN=01
                000860                        GN=01    EQU    *
                000860  58 20 C 020                    L      2,020(0,12)         GN=02
                000864  D5 00 6 000 6 025              CLC    000(1,6),025(6)     DNM=1-454    DNM=2-440
                00086A  07 82                          BCR    8,2
                00086C  58 10 C 004                    L      1,004(0,12)         PN=01
```

```
                 0009D4   07 F1                            BCR    15,1
156              0009D6                         GN=011     EQU    *
                 0009D6   F2 71 D 1D0 7 022                PACK   1D0(8,13),022(2,7)   TS=01       DNM=1-320
                 0009DC   F9 10 D 1D6 C 069                CP     1D6(2,13),069(1,12)  TS=07       LIT+1
                 0009E2   58 F0 C 04C                      L      15,04C(0,12)         GN=013
                 0009E6   07 AF                            BCR    10,15
157              0009E8   F8 71 D 1D0 6 01D                ZAP    1D0(8,13),01D(2,6)   TS=01       DNM=2-293
                 0009EE   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 0009F4   F8 11 6 017 D 1D6                ZAP    017(2,6),1D6(2,13)   DNM=2-207   TS=07
158              0009FA   58 10 C 048                      L      1,048(0,12)          GN=012
                 0009FE   07 F1                            BCR    15,1
159              000A00                         GN=013     EQU    *
                 000A00   F2 71 D 1D0 7 022                PACK   1D0(8,13),022(2,7)   TS=01       DNM=1-320
                 000A06   F9 10 D 1D6 C 06A                CP     1D6(2,13),06A(1,12)  TS=07       LIT+2
                 000A0C   58 F0 C 050                      L      15,050(0,12)         GN=014
                 000A10   07 AF                            BCR    10,15
160              000A12   F8 71 D 1D0 6 01F                ZAP    1D0(8,13),01F(2,6)   TS=01       DNM=2-329
                 000A18   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 000A1E   F8 11 6 017 D 1D6                ZAP    017(2,6),1D6(2,13)   DNM=2-207   TS=07
162              000A24   58 10 C 048                      L      1,048(0,12)          GN=012
                 000A28   07 F1                            BCR    15,1
162              000A2A                         GN=014     EQU    *
                 000A2A   F2 71 D 1D0 7 022                PACK   1D0(8,13),022(2,7)   TS=01       DNM=1-320
                 000A30   F9 11 D 1D6 C 06B                CP     1D6(2,13),06B(2,12)  TS=07       LIT+3
                 000A36   58 F0 C 054                      L      15,054(0,12)         GN=015
                 000A3A   07 AF                            BCR    10,15
163              000A3C   F8 71 D 1D0 6 021                ZAP    1D0(8,13),021(2,6)   TS=01       DNM=2-365
                 000A42   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 000A48   F8 11 6 017 D 1D6                ZAP    017(2,6),1D6(2,13)   DNM=2-207   TS=07
165              000A4E   58 10 C 048                      L      1,048(0,12)          GN=012
                 000A52   07 F1                            BCR    15,1
165              000A54                         GN=015     EQU    *
                 000A54   F8 71 D 1D0 6 023                ZAP    1D0(8,13),023(2,6)   TS=01       DNM=2-402
                 000A5A   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 000A60   F8 11 6 017 D 1D6                ZAP    017(2,6),1D6(2,13)   DNM=2-207   TS=07
167              000A66                         GN=012     EQU    *
                 000A66   F2 75 D 1D0 7 024                PACK   1D0(8,13),024(6,7)   TS=01       DNM=1-352
                 000A6C   FC 51 D 1D2 6 017                MP     1D2(6,13),017(2,6)   TS=03       DNM=2-207
                 000A72   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 000A78   F1 76 D 1D0 D 1D0                MVO    1D0(8,13),1D0(7,13)  TS=01       TS=01
                 000A7E   F8 33 6 009 D 1D4                ZAP    009(4,6),1D4(4,13)   DNM=2-95    TS=05
                 000A84   94 0F 6 009                      NI     009(6),X'0F'         DNM=2-95
169              000A88   F2 75 D 1D0 7 024                PACK   1D0(8,13),024(6,7)   TS=01       DNM=1-352
                 000A8E   FC 51 D 1D2 6 019                MP     1D2(6,13),019(2,6)   TS=03       DNM=2-233
                 000A94   F1 75 D 1D0 D 1D0                MVO    1D0(8,13),1D0(6,13)  TS=01       TS=01
                 000A9A   F8 33 6 00D D 1D4                ZAP    00D(4,6),1D4(4,13)   DNM=2-124   TS=05
171              000AA0   F8 73 D 1D0 6 00D                ZAP    1D0(8,13),00D(4,6)   TS=01       DNM=2-124
                 000AA6   FA 33 D 1D4 6 009                AP     1D4(4,13),009(4,6)   TS=05       DNM=2-95
                 000AAC   F2 75 D 1D8 7 024                PACK   1D8(8,13),024(6,7)   TS=09       DNM=1-352
                 000AB2   FB 33 D 1DC D 1D4                SP     1DC(4,13),1D4(4,13)  TS=013      TS=05
                 000AB8   F8 33 6 011 D 1DC                ZAP    011(4,6),1DC(4,13)   DNM=2-150   TS=013
                 000ABE   94 0F 6 011                      NI     011(6),X'0F'         DNM=2-150
174              000AC2   F8 73 D 1D8 6 011                ZAP    1D8(8,13),011(4,6)   TS=09       DNM=2-150
```

12

```
          000AC8   FD 51 D 1DA 6 015         DP     1DA(6,13),015(2,6)     TS=011        DNM=2-178
          000ACE   F8 73 D 1D8 D 1DA         ZAP    1D8(8,13),1DA(4,13)    TS=09         TS=011
          000AD4   D2 09 D 1E0 C 06D         MVC    1E0(10,13),06D(12)     TS2=1         LIT+5
          000ADA   41 10 D 1E6               LA     1,1E6(0,13)            TS2=7
          000ADE   DF 09 D 1E0 D 1DC         EDMK   1E0(10,13),1DC(13)     TS2=1         TS=013
          000AE4   06 10                     BCTR   1,0
          000AE6   92 5B 1 000               MVI    000(1),X'5B'
          000AEA   D2 07 6 0EF D 1E2         MVC    0EF(8,6),1E2(13)       DNM=3-439     TS2=3
176       000AF0   D2 08 6 083 7 000         MVC    083(9,6),000(7)        DNM=3-250     DNM=1-234
177       000AF6   D2 11 6 0C1 7 009         MVC    0C1(18,6),009(7)       DNM=3-293     DNM=1-260
178       000AFC   F2 75 D 1D8 7 024         PACK   1D8(8,13),024(6,7)     TS=09         DNM=1-352
          000B02   D2 09 D 1E0 C 06D         MVC    1E0(10,13),06D(12)     TS2=1         LIT+5
          000B08   41 10 D 1E6               LA     1,1E6(0,13)            TS2=7
          000B0C   DF 09 D 1E0 D 1DC         EDMK   1E0(10,13),1DC(13)     TS2=1         TS=013
          000B12   06 10                     BCTR   1,0
          000B14   92 5B 1 000               MVI    000(1),X'5B'
          000B18   D2 07 6 0D9 D 1E2         MVC    0D9(8,6),1E2(13)       DNM=3-329     TS2=3
179       000B1E   F2 71 D 1D8 7 022         PACK   1D8(8,13),022(2,7)     TS=09         DNM=1-320
          000B24   D2 03 D 1E0 C 077         MVC    1E0(4,13),077(12)      TS2=1         LIT+15
          000B2A   DE 03 D 1E0 D 1DE         ED     1E0(4,13),1DE(13)      TS2=1         TS=015
          000B30   D2 01 6 0E7 D 1E2         MVC    0E7(2,6),1E2(13)       DNM=3-383     TS2=3
180       000B36   D2 09 D 1E0 C 06D         MVC    1E0(10,13),06D(12)     TS2=1         LIT+5
          000B3C   41 10 D 1E6               LA     1,1E6(0,13)            TS2=7
          000B40   DF 09 D 1E0 6 009         EDMK   1E0(10,13),009(6)      TS2=1        .DNM=2-95
          000B46   06 10                     BCTR   1,0
          000B48   92 5B 1 000               MVI    000(1),X'5B'
          000B4C   D2 07 6 0F9 D 1E2         MVC    0F9(8,6),1E2(13)       DNM=4-0       TS2=3
181       000B52   F8 71 D 1D8 6 017         ZAP    1D8(8,13),017(2,6)     TS=09         DNM=2-207
          000B58   D7 05 D 1D8 D 1D8         XC     1D8(6,13),1D8(13)      TS=09         TS=09
          000B5E   94 0F D 1DE               NI     1DE(13),X'0F'          TS=09+6
          000B62   D2 03 D 1E0 C 07B         MVC    1E0(4,13),07B(12)      TS2=1         LIT+19
          000B68   DE 03 D 1E0 D 1DE         ED     1E0(4,13),1DE(13)      TS2=1         TS=015
          000B6E   D2 01 6 1C6 D 1E2         MVC    1C6(2,6),1E2(13)       DNM=4-54      TS2=3
182       000B74   D2 09 D 1E0 C 06D         MVC    1E0(10,13),06D(12)     TS2=1         LIT+5
          000B7A   41 10 D 1E6               LA     1,1E6(0,13)            TS2=7
          000B7E   DF 09 D 1E0 6 00D         EDMK   1E0(10,13),00D(6)      TS2=1         DNM=2-124
          000B84   06 10                     BCTR   1,0
          000B86   92 5B 1 000               MVI    000(1),X'5B'
          000B8A   D2 07 6 10F D 1E2         MVC    10F(8,6),1E2(13)       DNM=4-99      TS2=3
183       000B90   D2 09 D 1E0 C 06D         MVC    1E0(10,13),06D(12)     TS2=1         LIT+5
          000B96   41 10 D 1E6               LA     1,1E6(0,13)            TS2=7
          000B9A   DF 09 D 1E0 6 011         EDMK   1E0(10,13),011(6)      TS2=1         DNM=2-150
          000BA0   06 10                     BCTR   1,0
          000BA2   92 5B 1 000               MVI    000(1),X'5B'
          000BA6   D2 07 6 11A D 1E2         MVC    11A(8,6),1E2(13)       DNM=4-150     TS2=3
184       000BAC   FA 33 6 003 6 011         AP     003(4,6),011(4,6)      DNM=2-34      DNM=2-150
185       000BB2   FA 11 6 001 6 015         AP     001(2,6),015(2,6)      DNM=2-0       DNM=2-178
186       000BB8   D2 84 8 000 6 0A8         MVC    000(133,8),0A8(6)      DNM=1-428     DNM=3-201
          000BBE   58 10 D 1CC               L      1,1CC(0,13)            DTF=2
          000BC2   92 F0 8 000               MVI    000(8),X'F0'           DNM=1-428
          000BC6   58 F0 1 010               L      15,010(0,1)
          000BCA   45 E0 F 00C               BAL    14,00C(0,15)
          000BCE   50 20 D 1C0               ST     2,1C0(0,13)            BL =2
```

IA-02-02-22

```
                                    CROSS-REFERENCE DICTIONARY

DATA NAMES                      DEFN      REFERENCE

PERSONNEL-MASTER-FILE           00012     00133  00133  00138  00142  00142
IM01-SOC-SEC-NUM                00021     00176
IM01-NAME                       00022     00177
IM01-DEPARTMENT                 00023     00150  00198
IM01-NUMBER-DEPENDENTS          00025     00153  00156  00159  00162  00179
IM01-MONTHLY-PAY                00026     00167  00169  00171  00178
PRINTER                         00014     00133  00133  00138  00186  00192  00199
OR01-PRINTER-RCD                00030     00186  00186  00186  00192  00192  00192  00192  00199  00199  00199  00199
                                          00199  00199
WS-MASTER-FILE-EOF-SWITCH       00032     00135  00136  00145
WS-TOTAL-FAMILY-SIZE-SUM        00033     00185  00190  00194
WS-TOTAL-NET-PAY-SUM            00035     00184  00191  00194
WS-DEPARTMENT-HOLD              00037     00147  00147  00147  00150  00198
WS-FEDERAL-TAX-HOLD             00039     00167  00167  00171  00180
WS-FICA-TAX-HOLD                00041     00169  00171  00182
WS-NET-PAY-HOLD                 00043     00171  00171  00174  00183  00184
WS-FAMILY-SIZE-HOLD             00045     00174  00185
WS-TAX-RATE-HOLD                00047     00154  00157  00160  00163  00165  00167  00181
WS-FICA-TAX-RATE                00049     00169
WS-0-DEPENDENTS-TAX-RATE        00052     00154
WS-1-4-DEPENDENTS-TAX-RATE      00055     00157
WS-5-8-DEPENDENTS-TAX-RATE      00058     00160
WS-9-12-DEPENDENTS-TAX-RATE     00061     00163
WS-13-99-DEPENDENTS-TAX-RATE    00064     00165
WS-SWITCH-ON                    00067     00136  00145
WS-SWITCH-OFF                   00069     00135
WS-COLUMN-HEADING               00071     00199  00199
WS-DETAIL-LINE                  00090     00186  00186
WS-SOC-SEC-NUM                  00093     00176
WS-NAME                         00096     00177
WS-MONTHLY-PAY                  00099     00178
WS-NUMBER-DEPENDENTS            00102     00179
WS-PAY-AVERAGE                  00105     00174
WS-FEDERAL-TAX                  00108     00180
WS-TAX-RATE                     00111     00181
WS-FICA-TAX                     00114     00182
WS-NET-PAY                      00117     00183
WS-FOOTING-LINE                 00120     00192  00197
WS-TOTAL-FAMILY-SIZE            00126     00190
WS-TOTAL-NET-PAY                00129     00191

PROCEDURE NAMES                 DEFN      REFERENCE

0010-READ-AND-PRINT             00141     00136
0010-EXIT                       00188     00136  00146
0020-TOTALS-RTN                 00189     00144  00151
0020-EXIT                       00196     00144  00151
```

```
28/08/78   PHASE  XFR-AD  LOCORE  HICORE  DSK-AD   ESD TYPE  LABEL     LOADED  REL-FR

       PHASE***  005000  005000  006213  2E 12 4  CSECT     DUMPOPE   005000  005000

                                                  CSECT     IJGFIEWZ  005E28  005E28
                                                  *  ENTRY  IJGFIZWZ  005E28
                                                  *  ENTRY  IJGFIZZZ  005E28
                                                  *  ENTRY  IJGFIEZZ  005E28

                                                  CSECT     ILBDSAE0  0060D0  0060D0
                                                     ENTRY  ILBDSAE1  0060F0

                                                  CSECT     IJDFAPIZ  005D10  005D10
                                                  *  ENTRY  IJDFAZIZ  005D10

                                                  CSECT     ILBDMNS0  0060C8  0060C8

* UNREFERENCED SYMBOLS                            WXTRN     STXITPSW
                                                  WXTRN     ILBDDBG2

002 UNRESOLVED ADDRESS CONSTANTS
```

SSN	NAME	MONTHLY PAY	NO OF DEPS	PAY AVE	FED TAX	FICA TAX	NET-PAY

```
0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 005AC8 - CONDITION CODE 2 - DATA EXCEPTION
0S60I JOB DUMPPE2  CANCELED





         DUMPPE2           28/08/78                                                              22.46.02                  PAGE   1

GR 0-7   0C005CA7 0CC05AA6 C00C54C0 CC005000    CC0052RP 50005CCE 000050E8 0000530A
GR 8-F   CCCC54CC C0C05C9E 000C50C0 00CC5000    CCCC576P 0000554R 80005C74 00005A54
COMREG   RG ADDR IS C0036R

0CCCC0   0CCCCCCC CCC30000 00C04C6A C2000438    CC0C0030 00000368 FF050030 00000000    ................ ................
000020   FF05CC0C 40004884 FF150007 F0CC5ACE    5858C2C5 D6D1F3C1 FF150130 80005DC2    .... ........... $8BEOJ3A.......8
0CC04C   000C1F5C CCC00000 00001*2A CCCC0000    F6F83300 076E29FC 00040000 0F0023D2    ...&............ 6R..............K
000060   0C040CC0 CC0376F4 0CC4C000 C0002392    CC000030 000011E4 00040030 030395A4    ........4....... ........U.......
0C0C80   58F2F6F2 C6C2C8F6 C000C000 CCCC0003    CCC5CC03 000306RC 06800680 4188C073    $262FBH6........ ................
0000A0   457CC148 540FC393 41A0C558 4570E1R0    18A84190 017R4180 C1A047F0 000C0680    ......C...O..... .........A..O...
0000C0   068C068D C6P00680 0680C6PC 41P80017    41880050 4570014R 41800178 9640A001    ................ ...&......... ..
0CC0E0   9170A00C 47100CEC 9260A001 95E2A002    4780P48A 95C1A002 4780849A 9561A002    .........-...S.. ......A......./..
000100   47R0011C 91C4A000 4780C11C 9203CC8F    9281A030 4BA003C2 487AD604 49A003D6    ................ ........B..O....0
0C0120   41AAC64P C77894F9 703RC701 705F7C58    9283A000 96R0A001 440009EA 0788947F    ..O.....9..P.... ................
0C014C   ACC14570 CC9AC7FA 42R0C0E5 58E002P8    9180A000 07175890 A0044890 02225000    ........B...Z... ...............&.
000160   902CD218 90100298 D2C7900R 8CCC5680    AC0C07F7 45700150 02078A32 8A3858E0    ..K......K...... ...7....6K......
0C01RG   02C49120 P77A47R0 C1PF92P5 C6A8CCC7    RA32C64R 1PAADD0R 8A32000C 43A1C009    .D...........n.. ..O.............
0C01A0   42A00357 41AAD648 0200C589 03579540    05894740 01CC9560 05894790 01C69240    .......O.K...... ... ...-.....F.
0001C0   058547F0 01CCD200 05P9C5RR 459002CA    47FCC662 5890A0C4 4220A000 9140A001    ...O..K......... .O............ ..
0C01EC   471CCRC6 C20702*8 9CC84PA0 03C241AA    C6040201 0016A000 98989010 82000298    ...FK.........B.. O.K.............
000200   459002CA 4400A004 5890A004 98C8902C    91020021 4780021C 9200058A 989F0298    ................ ................
000220   820CC020 5784D648 91E0877A 47800262    9120877A 47100254 9180886A 4710C254    .......O........ ................
0C0240   9103R778 47700178 9503C093 478C0179    47F00262 02070298 8AC09890 8AC88200    ................ .O..K.........H..
0C0260   0258D207 025R8A98 9890EAA0 82000298    5680A030 41100030 47F08F8E 96030039    ..K............. ..........O......
0CC2PC   5601CC38 82000038 CC615C40 61504044    FCCC0030 00000000 FF050133 A0004802    .........../* /& . O...............
0002A0   00038FA8 90C38094 0C037D88 0303AE70    C0034C8C 80001DAC 00001000 00002000    ................ ................
0C02C0   CCCC3CCC CCC04000 36D2C500 05R9058A    C789906R 05A09878 0C505970 8A2447D0    ....... ..KN.... ..........&......
0C02E0   02EA8A70 CCC847F0 02EFF870 CCC81F78    1F871C78 5E7005AC 5080058C 9130058A    ........O....... ..........&......
000300   4770034C 12774740 03401866 4A7CC580    5C600504 406005R0 588005CC 18675E60    ........ . ...... .-.. -.........-
000320   F0205C6C F0205880 05C81867 5E608020    5C6C8020 588005C4 5E708020 50708020    ..&-.....H...-.. &-.....D....&...
0C0340   D20005PA 05R99R68 C5ACC7F9 1864360    C58A9E6C C00258P6 C5C05E70 801C5070    K...........9...- ...-.............&.
0C0360   801C47F0 C340C000 F2F861F0 FR61F7F8    5CGC5000 00000000 00000030 00000000    ...O. ..28/08/78 &.&..............
0C0380   C4E4D4D7 C7C5F24C CC01C7FF 0C006213    00C06213 00000010 0003FFFF DAFD4E80    DUMPPE2 ..P..... ...............+.
0C03AC   28AC7EC0 0C6337C9 37C03901 39923A5C    3C703C74 3C7838F2 F8F0F8F7 F8F2F4F0    ........I.......* ........280878.40
0003C0   000C3648 CCCG0014 1P6C1996 1A3C1.4C    1A5CC020 378C0010 5858C206 00060007    ..........-...... .*........$$BO....
0C03E0   01001R04 3604C000 CG000000 036P1080    CG0003F4 00000000 00003E7C 00000044    ................ ...4............
CC040C   0C001A74 CCCOG000 0C0CC000 CCCCCCC0    CCCC0000 00000000 00000000 00000000    ................ ................
0C0420   0C0C0300 CCC04500 CC00C5A0 CCC03F38    CCCC0CCC 00C00000 92380223 909FC298    ................ ................
0C0440   5RRF02RP 47F00C1C 47F0C646 4AA003C2    9588A000 47800468 9180A300 4710C47A    ......O...O.....8 ................
000460   58R0ACG4 5CC8R02C 58PEC288 5C65C5A0    459004EC 986905A0 07F99601 A000988E    ................ ..........9......
0004R0   02889C69 C5A04590 04E09969 05A095FF    A00FC789 91A0877A 478004C2 9640877A    ................ ..............B. ..
0C04AC   547FP77A C21RPAC8 C29850C0 8AE449E0    C222D207 8AC0E000 94F08AC1 988E0288    ....K..H..&..U.. ..K...........A....
0004C0   07F95000 PAPC4PE0 0722C207 8A58E030    5RE002C4 94FD8A99 D2188A40 C29807F9    .9&........K.... ...D.....K.......9
0004EC   D2C00589 C58A9P78 0C5C5970 PA2447C0    04FA8A70 000847F0 04FE8970 03081F78    K.........&..... ........O........
0C05CC   1P871C7P 5F7005RC 9180A000 4780051E    55FFAG0F 4780051E D2000588 058A9130    ................ .........K.......
0C0520   C5RA477C C5729140 05RAC719 508005PC    12774740 056C1866 4A700592 506005P4    ........ ....&.. ... ..........-..
000540   406005R2 5880035CC 18675E6C 90245060    8C245880 05C81867 5E608024 50608024     -..........-..&- ......H...-..&-..
0C0560   5RAC05C4 5E708024 5C7C8024 9240058A                      5080058C 19664360    ...D....&.... .. .9.......&.......-
0C05RG   058A8F6C C0025R8A 05CC5F70 8C1C5C70    IA-02-02-2214     9240055A 07F90000    ...-...........&. ........... ...9..
```

```
          DUMPPE2            28/08/78                                                                                   PAGE  11

0043AC    0003A06C C0000030 0003FFFF CAFD4E80    20004E8C 006337C5 37003991 39923A5C    ...............*.  ..*....I.......*
0043CC    3C7C3C74 3C7938F2 F8F0F8F7 F8F2F4F0    C0003A48 00000014 19201996 1A3C1A4C    .......280878240   ................
0043EC    1A5C0070 378C0C30 CCCCC000 CCC0C0C0    CCCCC0C0 3C0400CC 00000000 03680080    .*..............   ................
C044CC    00004444 CCC0C000 00003E7C 0CC0C044    C0001A74 0000000C 00000000 0C000000    ................   ................
004420    0C00CCCC CCCCC000 CC0CC000 CCCC0000    CCC00000 0000000C 00000540 00004030    ................   ............... .
004440    0C000000 0C000000 0C003E7C CCC00044    CC001A74 0000000C 00000C00 00000000    ................   ................
00446C    CCCCCCCC CCC00C00 0CCCC000 CCC0C000    CCCC0000 00000000 00000540 00004128    ................   ................
004480    CCC00C00 --SAME--                                                               ....
004500    C3D9F00C CC1C000F C3D7F000 C010000F    C7D9F000 E01F000F C4D2F013 0130130F    CRO.....CPO.....   PRO.....DKO.....
004520    C4D2F113 1131131F C4D2F213 2132132F    C4D2F313 3133133F C4D2F413 4134134F    DK1.....DK2.....   DK3.....DK4.....
004540    C4D2F514 0140140F C4D2F614 1141141F    C4D2F714 2142142F E3D7F018 0180180F    DK5.. ..DK6.....   DK7.....TP0.....
00456C    E3D7F118 1181181F E3D7F218 2182182F    E3D7F318 3183183F E3D7F418 4184184F    TP1.....TP2.....   TP3.....TP4.....
004580    E3D7F518 5185185F E3C4F018 0180180F    C5D5C44C 40000000 C5D5C440 40004500    TP5.....TD0.....   END  ...END  ...
0045AC    CCCCCC00 --SAME--                                                               ....
0C4760    5B5BC2C4 E4D4D7C2 5B5BC2C4 E4D4D7C2    CCC0C001 00000006 06040309 40404040    $$BDUMPB$$BDUMPB   ...........Q
004780    40404040 --SAME--
004BCC    4C4C4C4C 4C4C4040 40404040 40404040

LALTYP    HEX LENGTH IS 0000
--BG--
004FA0                      D5D64005 C1D4C540    FF150007 E0005ACE 00005C9E 00005000            NO NAME   ..........*...&.
004FCC    0C005C00 00005768 00005548 80005C74    C0005A54 00005C82 00005A66 000054C0    ..&...........*.   ......*.........
004FEC    0C00500C CC0052B8 50005CCE 0CC050E8    000053D8 000054C0 00000000 01767F86    ..&.....&.*...&Y   ...Q............
005CC0    05F00700 900EF00A 47F0F0A2 CC0C5000    CC005000 0001D7FF 00005000 0001D784    .0....0..00...&.   ..&...P...&...P.
0C5020    FFFFFF7C CCC05000 00004FB8 000C6870    0A0407F1 00005010 00005010 00005FA8    ......&.........   ...1..&...&.....
005040    000C6FA8 CC000368 D202733E 7299F912    7341733E 4740703A FA1C7348 73CFF911    ........K.....9.   ..... ........9.
005060    73467348 47807014 D2117168 72760202    7187728A D2087192 726D0202 71A37299    ........K.....K.   ....K.....K.....
0C5080    D28372BA 58C0F0C6 58F0C000 58D0F0CA    55C0E000 4770F0A2 9610D048 92FFE000    K.....0F......0.   ......0.........
0050AC    47F0F0AC 58CEFC8A 9CECC00C 185C589F    FCBA9110 D0480719 C7FF070C 00005C9E    .00...0.........   0.............*.
0050C0    00005000 CC005000 0000576A CC005548    CC005800 00005C84 C3D4C2C6 F0F0F0F1    ..&...&.........   ......*.CDBF0001
0050F0    C4E4D4D7 F0C7C540 F1C6C9D2 C9C5C7F0    F1000454 4C000329 4C004896 2CE24000    DUMPOPE 10RKINGO   1............S .
005100    8C058C20 CC150C12 0CC8CC05 0CF0F1F0    40404040 40404040 40404040 4040E2E2    .............010                 SS
005120    D5404040 40404040 40404040 40404040    D5C1D4C5 40404040 40404040 404040D4    N                  NAME            M
005140    C6D5E3C8 D3E84CD7 C1E84040 C5D64CD6    C640C4C5 D7E240D7 C1E540C1 E5C54040    ONTHLY PAY  NO O   F DEPS PAY AVE
005160    40C6C5C4 40E3C1E7 40404040 C6C9C3C1    40E3C1E7 404040D5 C5E340D7 C1E84040     FED TAX    FICA    TAX   NET-PAY
00518C    40404040 40404040 4040E3C5 404040C6    4C404C40 40404040 404040C5 E340D7C1               TE   F                ET PA
0C51AC    E8404040 4C40404C 4C404040 4040D7C1    40404040 40404040 40404040 40404040    Y             PA
0051CC    40404040 --SAME--
0051EC    4C4C4040 40404040 404C4040 40404040    406C4040 404040D9 4C404040 40404040                          .    R
005200    404C4040 4C404040 40404040 40404040    4C4C4040 404040C8 40404040 40404040                               H
005220    404C4040 4C404040 40404040 40404040    4C404040 40404040 404040C4 C5D7C1D9                                      DEPAR
005240    E3D4C5D5 E340E3D6 E3C1D340 C6C1D4C9    D3E840E2 C9E9C540 4040C4C5 D740C1D5    TMENT TOTAL FAMI   LY SIZE   DEP AN
005260    C440E3D6 E3C1D340 D5C5E340 D7C1E840    4040D3E8 40E2C9E9 C5404040 40404040    D TOTAL NET PAY      LY SIZE
005280    404C4C40 --SAME--
0052A0    CC0CCC5C CCC0CC00 CCCCC000 00CC0000    CCCC0000 00000000 00008234 0C00C106    ...&............   ................
0052CC    00005320 00005340 04005F28 200EE2F8    E2F0F0F6 40010000 00000000 00028020    ....... ......SY   S006 ...........
0C52EC    CCCCCCCC C4C05308 8CCCCCC0 CCC0C046    CCC00000 00460000 02005904 00C80013    .......Q........   ..............H.
005300    2080004F CC4A0000 010040F0 5821CC58    CC0053D8 00000050 00005427 8E0060D0    ..........-0....   ...Q...&......-.
0C532C    C70C52F2 4C00C0C6 310052F4 4CC00005    CA00532B 00000000 060053A0 00000050    ...2 ......4 ...   ...............&
005340    28000C4F CCC00000 0C0C8000 CC000105    CCC05370 00005378 00005D10 088488F1    ................   ...............1
005360    010054C1 C0C00000 07004120 ECC0000C    01005431 20000084 06005418 00000050    ...A............   ...............&
0C5380    F6F3F9F1 F6F7F0F3 F9C8D6E3 E340D9D6    C2E84040 40404040 404040F0 F1F0F1D7    639167039HOTT RO   BY          0101P
0053A0    C6C3F0F9 F0F3F9F8 F0F04040 40404040    40404040 40404040 40404040 40404040    FC09039800
0053CC    40404040 4C4C4C4C 40404040 40404040    0C1C58E0 F06407FE F7F8F4F7 F6F4F0F2                       ....0...78476402
```

```
         DUMPPE7              28/08/78                                                                        PAGE  12

0053E0   F5C7D9E4 D5C4E840 C7C1D9E5 C5E840D1    D94040F0 F1F0F1E2 C6C3F1F2 F0F5F6F8    5GRUNDY GARVEY J  R  0101SFC120568
005400   F0F04040 40404040 40404040 40404040    40404040 40404040 40404040 40404040    00
005420   40404040 40404040 E00C4770 F05A47F0    F1404040 40404040 40404040 4040E2E2            ....0..0  1             SS
005440   D5404040 40404040 40404040 40404040    D5C1D4C5 40404040 40404040 404040D4    N                 NAME           M
005460   D6D5E3C8 D3E840D7 C1E84040 D5D640D6    C640C4C5 D7E240D7 C1E840C1 E5C54040    ONTHLY PAY  NO O  F DEPS PAY AVE
005480   40C6C5C4 40E3C1E7 40404040 C6C9C3C1    40E3C1E7 404040D5 C5E360D7 C1E84040     FED TAX    FICA   TAX   NET-PAY
0054A0   40404040 40404040 40404040 40404040    40404040 40000000 47F0F04C 0A320000                            ....00.....
0054C0   0A320000 47F0F01A C9D1C4C6 C3E9E9E6    F3F99190 10024710 F0240A07 90CEF060    .....00.IJDFCZZW  39......0.....0-
0054E0   58F01018 48C0102E 06C018E0 44C0F06C    58CEFC60 D2001028 1017D200 10171016    ..............0.  ..0-K.....K.....
005500   0ACC07FE 41801002 4710F0F6 CAC74200    1C280400 07FE0000 00000000 0C000000    ..........0.....  ................
005520   00000000 D203F000 C000F01A CA32C000    C9D1C3C6 E9C9E9F0 030A0A00 91801002    ....K.....0.....  IJCFZIZ0........
005540   471CFC26 CA075CE0 CC645BFC 10209101    10044780 F04C914C 10024710 F04658E0    ..0...&.........  ....0.. ....0...
005560   F04447F0 FC1458E0 101CC7FE D801FC40    FCCC4770 F05A47F0 F04658E0 F06407FE    0..00.......N.0-  ....0..00...0...
005580   615C0CCC CCC00000 00F0F04C 0A320000    3000048 00000000 00000000 00005800     /*.......00.....  ................
0055A0   CCCCCC0C CCCCC000 F024CA07 90CEFC60    58E01018 48C0102E 06C018D0 44C0F06C    ........0.....0-  ..............0.
0055C0   58CEFCAC C2001028 1017C200 10171016    CCCC0C48 00000000 000000C0 00000000    ..0-K.....K.....  ................
0055E0   CCC00CCC CC000000 C000C000 00CCCC00    CCC00000 00000000 00000000 00000000    ................  ................
005600   CCCCC000 --SAME--                                                                ....
0056E0   CC0C0CCC CC000000 0CCCC000 0CCCC0C0    CC000000 0000000C 00005000 0C00C000    ................  ..........&.....
005700   0CCCCCCC CCC0530B CCCC54C0 CCCC5CE8    CCC052B8 C0005348 00000000 000783AC    .......0......&Y  ................
005720   0C0CCC0C 004B962C CCCCC000 CC000000    CCCCC00C 00000000 00000000 00005BDC    ................  ..............*.
005740   0CCC500C CC005C92 CC00C470 CCCC0CE8    CCCC5960 00005C4E 00005C92 00000DE8    ..&...*........Y  ...-..**..*....Y
005760   0CC003C8 C0C0035A 0CCCE0CA CCCC58C4    CCCC58D6 00005B0C 00005C4E 00005C7C    ..........-H...M  ..*0..*...**..*.
005780   CC005CB2 CC005860 CC005872 CCCC59C4    0CC0592E 0000591A 00005944 00005956    ..*....-........  ................
0057A0   000059AC 00005986 00005944 CCC054D6    CCCC5A66 00005400 00005A2A 00005454    ...............0  ................
0057C0   CC0C5C3C CCC058DC CC0C5C4E CC005CA2    CF5C9C01 3C40205B 202C2120 4B202040    ..*...*...**..*.  .*... .*.......
0057E0   202120F0 2C212000 0C0CC000 0CC0C004    5B5BC2D6 D7C5D540 5B5BC2C3 D3D6E2C5    ...0............  $$BOPEN $$BCLOSE
005800   5810D1C8 4810C0B6 94BF1000 4110CCB8    5800D1C8 1A4005FC 5C00F008 4500F00C    ..JH............  ..JH. .0&.0...0.
005820   CC0052B8 CA024110 C0885800 D1CC1840    C70C05F0 5000F008 4500F00C 00005348    .............J..  ...0&.0...0.....
005840   0A025020 D1C05880 D1C0D200 60C06026    5800D208 5000D1F4 5800C01C 5000D208    ..&.J...J.K.-.-.  ..K.&.J4....&.K.
005860   5820C020 D5C060C0 6025C7B2 5810C004    C7F15800 D1F4500C D2058A00 D1C81840    ....N.-.-.......  .1..J4&.K...JH.
005880   4110C09C C70005FC 5C00F0C8 45C0F0CC    CCCC0C00 0A025810 D1C84810 C08605F0    .......0&.0...0.  ........JH.....0
0058A0   5180C1C0 4780FC16 41101004 412010E0    D2DF100C 20005800 D1CC1840 4110C090    ......0.........  K.......J.. ....
0058C0   C70005FC 5C00F008 45C0F00C CCCCC000    0A020A0E 5810D1CA 58F0C024 91201010    ...0&.0...0.....  ......JH.0......
0058E0   C71F1841 41F0C024 D2C21041 FCC158F0    1C1C45E0 F0085020 D18C5870 D18C58F0    .....0..K...0..0  ....0.&.J...J..0
005900   C02807FF 58C0020C 5000C1F8 5800CC2C    5CC0D20C 5810C00C 07F15800 D1F85000    ......K.&.JB....  &.K......1..JB&.
005920   D20CD2CC 60006025 5810C008 07F15820    C03C954C 60070772 D5006008 60070772    K.K.-.-......1..  ... -...N.-.-...
005940   58CCD21C 5C00D1FC 5800C034 5C00D210    581CCC10 07F15800 D1FC5000 D2105810    ..K.&.J......&.K.  .....1..J.&.K...
005960   C03807F1 5820CC38 D5C160C7 7018C792    58C0D20C 5000D200 5900C03C 5000D20C    ...1....N.-.....  ..K.&.K.....&.K.
005980   5810C00C C7F15800 D20C5000 D20C5800    D2105000 D2045800 C0405000 D2105810    .....1..K.&.K...  K.&.K.... &.K...
0059A0   C01007F1 5800D204 5000D210 F271D100    7C22F910 D1D6C0A8 58F0C044 077FF871    ...1..K.&.K.2.J.  ..9.JO...0....8.
0059C0   D1D06018 F176D100 D100F811 6017D1D6    5810C048 07F1F271 D1D07022 F910D1D6    J.-.1.J.J.8.-.JO  .....12.J...9.JO
0059E0   C06558F0 CC4C07AF F871D100 6010F176    D1D0D1D0 F8116017 D1D65810 C04807F1    ...0....8.J.-.1.  J.J.8.-.JO.....1
005A00   F271D1D0 7022F910 D1D6C06A 58F0C050    C7AFF871 D1D0601F F176D100 D100F811    2.J...9.JO...0.&  ..8.J.-.1.J.J.8.
005A20   6017D1D6 5810C048 C7F1F271 D1D07022    F910D1D6 C06558F0 C05407AF F871D100    -.JO.....12.J...  9.JO...0....8.J.
005A40   6071F176 D100D100 F8116017 D1D65810    C048C7F1 F871D100 6023F176 D100D100    -.1.J.J.8.-.JO..  ...18.J.-.1.J.J.
005A60   F8116017 D1D6F275 D1D07024 FC51D1D2    6017F176 D1D0D1D0 F176D100 D100F833    8.-.JO2.J.....JK  -.1.J.J.1.J.J.8.
005A80   6009D1D4 94CF6009 F275D1D0 7024FC51    D1D26019 F175D100 D100F833 6000D1D4    -.JM..-.2.J.....  JK-.1.J.J.8.-.JM
005AA0   F873D1DC 60C0FA33 D1D46005 F275D1D8    7024F833 D1CCD1C4 F8336011 D1D8940F    8.J.-...JM-.2.JQ  ....J.JM8.-.J...
005AC0   6011F873 D1D86011 FD51D1DA 6015F873    D1D8D10A D209D1E0 C06D4110 D1E6DF09    -.8.JQ-...J.-.8.  JQJ.K.J......JW..
005AE0   D100D1DC CA10925B 1000D207 60EFD1E2    D2C86083 7000D211 60C17009 F275D1D8    J.J....$..K.-.JS  K.-...K.-A..2.JQ
005B00   7024D209 D1F0C060 4110D1E6 DFC9D1E0    D1DC0610 925B1000 D2076009 D1E2F271    ..K.J.....JW..J.  J....$..K.-RJS2.
005B20   D1D87022 D20301F0 C077CE03 D1E0D1DE    D20160E7 D1E2D209 D1E0C060 4110D1E6    JQ..K.J.....J.J.  K.-XJSK.J.....JW
005B40   DF09D1EC 60C90610 925B1000 D2076CF9    D1E2F871 D1D86017 D705D1D8 D1D8940F    ..J.-....$..K.-9  JS8.JQ-.P.JQJQ..
```

```
DUMPPE2          28/05/78                                                                                    PAGE  13

005B60   D10ED203 D1F0C078 CF03C1F0 D10ED201   F106F1E2 D209D1F0 C0604110 D1E6CF09   J.K.J.....J.J.K.  /.JSK.J.....JW..
005B80   D1E06C0D C6109258 1000D207 610FD1F2   F2C9D1F0 C0604110 D1E6DF09 D1E06011   J.-....$..K./.JS  K.J.....JW..J.-.
005BA0   C61C9258 10C0D2C7 611AD1F2 FA3360C3   F011FA11 60016015 D2848000 60A85810   ...$..K./.JS..-.  -...-.-.K...-...
005BC0   D1CC92FC 8CC058F0 1C1045E0 FC0C5C20   D1CC5890 D1C05810 D20807F1 D203C1F0   J..0...0....0.&.  J...J...K..1K.J.
005BE0   C077CF03 D1E06001 D2026172 D1F1F873   D1D96003 D703D108 D1D8940F D1DCD209   ....J.-.K./.J.8.  JQ-.P.JOJQ..J.K.
005C00   D1E0C06D 4110D1E6 DFC9D1F0 D1DCC41D   92581CCC D20761FA D1E2D283 80006130   J.....JW..J.J...  .$..K./.JSK.../.
005C20   924080C4 5810D1CC 926C80C0 58FC1010   45F0F00C 5020D1C0 5880D1C0 D2016001   . ....J..-...0..  ..0.&.J...J.K.-.
005C40   C07FD2C3 6C03CC81 5810D20C 07F1D201   6CC77018 D2798000 60289240 807AD209   ..K.-.....K..1K.  -...K...-.. ..K.
005C60   8C798074 5910D1CC 97F1E00C 58FC1010   45E0F00C 5020D1C0 5880D1C0 5810D210   ......J..1...0..  ..0.&.J...J...K.
005C80   C7F10ACE 5CD05CC8 5C50C004 582CC0C0   95C07000 077992FF 20039610 D04850E0   .1..&.&.&&......  ...............&.
005CA0   F05405FC 9120D048 47F0F016 58C0FC48   5820FC5C 58E00054 07FE9620 D0484160   ...0......0.....  ...&...........-
005CC0   00044110 CC044170 CC69C670 05505840   1000E418 50401000 87165000 4180018C   ..............&.  ....& ....&...J.
005CE0   4170D1CF C8105800 8CCC1E08 5CCC8CCD   87861C0C D208D208 C05C5860 D1C45870   ..J.........&...  ....K.K..*.-JD..
005D00   D18C5880 D1C058E0 C05407FE 7CC096F0   0A320000 47F0F084 0A320000 47F0F01A   J...J..........0  .....00.......00.
005D20   C9D1C4C6 C1D7C9E9 F3F9918C 10024710   F0240A07 90CFF0EC 58E01018 06E00200   IJDFAPIZ39......  0.....0.......K.
005D40   1017FCC0 18CC18DC 43C01017 41E0C010   43CEFCF7 19DC4780 F0504620 F0400A32   ................  ..07....0&..0 ..
005D60   43DEF107 43C01016 19CD4770 FC769180   10164790 F076920B 10280A00 91801002   ..1.........0...  ....0...........
005D80   4710FC76 0AC742D0 1C2842D0 1C16CAC0   91801002 4710F08A 0A07D201 10261003   ..0.............  ......0...K.....
005DA0   92011C28 58E01028 D2021029 10195CE0   1C180650 98CDF0EC 44001022 58E0F0F4   .........K.....&.  .......0.......04
005DC0   0A00C7FE 91801002 4710F08E 0AC79101   1C15479C F0D494FE 10159102 1026C78E   ..........0.....  ....0M..........
005DE0   47F0F0DA 91C11027 078E1200 4780F0E4   18F007FF 92881028 0A0007FE 00005768   .00...........0U  .0..............
005E00   C0005548 FCC05C74 F2C2C1F9 F8F7F6F5   F4F3C3F1 4E60F040 9308D3C8 C388B3A8   .......*.2BA98765  43C1+-0 ..L.C...
005E20   A399F388 C3181308 47F0F014 47F0F252   47F0FC22 47F0F014 47F0F014 0A32C901   ..T.......00..02.  .C0..00..00...IJ
005E40   C7C6C9C5 E6F9F3F9 0C0890AE F26C9180   10494710 F04A58A0 102C5080 10584AA0   GFIEWZ39.....2...  ....0......&....
005E60   104A50A0 10609140 10274710 F1CC47F0   FCAC598D 105896BC F088508D 10589110   ..&..-. ....1..0  0.......0.&.....
005E80   1015478C FC7E18D0 41ACC100 18CA47C0   FC76D2FF 80008000 1ADA868A F06441CC   ....0...........  0.K...........0...
005EA0   00FF44C0 F25A98AE F26C4400 105407FE   91081049 4780F098 94F71049 47F0F1CC   ....2...2.......  ......0..7...01.
005EC0   91081064 4710F0A2 0A0C9180 10024710   F0AC0A07 91C11004 4780F098 47F0F236   ......0.........  0.........0..02.
005EE0   91081003 4780F0C4 91801064 4780F0D0   58E01064 47F0F0F4 91921003 4780F110   ......0D.......0.  .....00U.......1.
005F00   91801064 4780F14C 58F01064 98ADF26C   5CF1F100 58101090 070005EE 98F1E00C   ......1.......2.  .11...........1..
005F20   90ACF26C 47F0F154 CCCCC000 CCCCCCC0   CCCCCC00 C000000C 91401005 4780F158   ..2..01.........  .......... ....1.
005F40   48810000 4990F260 4780F13C 4880105E   4980F260 4720F124 4740F13C 96401049   ......2-..1.....  ..2-..1.. 1.. ..
005F60   47F0F158 91041064 4780F0C8 58E01C50   47F0F0E4 91201064 4780F158 96101049   .01........00...&  .00U......1.....
005F80   5880108C 4188C000 50801058 914C1049   4710F178 4A80104A 50801060 47F0F18A   ........&.... ..  ..1......&..-.01.
005FA0   48C0104A 48C10000 1AC850C0 10609740   10494BE0 103499E0 00084350 F26341B0   .....A....&..-.  ..............2...
005FC0   000518CC 43C8103B 43C81043 19CD4770   F18A47E8 103B8AEC CC084680 F19C41CC   ................  1.............1...
005FE0   000142C8 103BD503 103C1036 4700F202   96C81049 9001F280 98ADF26C 4100F284   ......N.......2.  ......2...2...2.
006000   4110F264 0AC29801 F28C5120 10104710   FC7F9140 1027478C F1FA948F 102747F0   ..2......2.......  0.. ....1.......0
006020   F0AC91C8 1C494710 F21A9108 10644780   F21A58B0 10800202 10811020 5080102C   0........2.......  2.....K......&...
006040   0A009110 10494780 F22A9710 104947F0   FCA89680 104998B0 105847F0 F0569180   ........2.......0  0.............00...
006060   1026471C F1CC9120 1C274710 F1CC94FF   1C495640 102747FC F1CCD203 10581080   ....1.......1...  ... ....01.K....-
006080   C7FED20C DCC08C00 CCC0C001 585BC2D6   D7C5D540 00005000 0C005000 00005768   ..K.........$$BO  PEN ..&...&.....
0060A0   00005548 80005BF6 C0000000 CC000000   FFCC0000 000000CC 00000000 00000000   ........6.......  ................
0060C0   CCCCCC00 CC00C7FE FFC81049 4780FC58   C5FC47F0 F018C9D3 C2C4E2C1 C5F0F3F1   ...............0.  .0.00.ILBDSAE031
0060E0   F4F3F0F3 61F0F761 F7F41800 47F0F022   41C0C001 05F050E0 F0EE9005 F0F24144   4303/07/74...00.  .....0&.0...02..
006100   C0CC9110 40144780 F03C91C8 40154710   F0309120 40024780 FC2C4100 0C1047F0   .... ...0... ...  0... ...0.......0
006120   F03C5811 C0044154 CCC04850 F11A5855   CC0C4155 00001255 4790F0AC 18350501   0...........&1...  ...........0...N.
006140   F0E43000 47F0F05A 41330002 47F0FC48   5C50F0EA D5033006 F0F04780 F09E4403   0U....0.......00.  &&0.N...0...0...
006160   CCC644C8 C0CA1852 92F05000 D2065C01   5C001200 4780F09A 4900F056 4780F092   ..........0&.K.&.  &.....0...0M..0.
006180   96015C02 47F0F09E 96015001 47F0FC5E   56C15000 58E0FCEF 9805F0F2 59F0F0EA   ..&..00....&..00.  ..&...0...02.00.
0061A0   07FF85C0 C0184840 F11C1604 4110F10A   0A02910F D0494780 F0DC5910 F11258F0   ....... 1.....1.  ........0...1..0
0061C0   F116C1C1 1CC21002 41110D3C 5CD10CC0   C7FF1900 0A060CCC 000005FC CC01103B   1.P.........&J..  .............0....
0061E0   43C81C43 19CD4770 F1B842F8 103B88E0   CCC84680 F19C41CC 000142C8 103BD503   ........1.......  ....1.........N.
006200   5B5BC2C3 C6C2C5D9 C0C0C000 CCC00000   C0040008 00000000 00000000 00000000   $$BCOMER........  ................
```

IA-02-02-22

DEBUG

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 4

PROGRAM NAME: DUMP-PE

```
      1                               IBM DOS AMERICAN NATIONAL STANDARD COBOL                    CBF CL3-4       28/09/78

   CBL LIB
  00001            BASIS C320MP
  00002    000010 IDENTIFICATION DIVISION.
  00003    000020 PROGRAM-ID. COMP-PE.
  00004    000030 AUTHOR. LT HUDGIN.
  00005    000040 REMARKS. PE DESIGNED FOR USING ALC TO DEBUG A COBOL PROGRAM.
  00006    000050 ENVIRONMENT DIVISION.
  00007    000060 CONFIGURATION SECTION.
  00008    000070 SPECIAL-NAMES.
  00009    000080     C01 IS CHAN1.
  00010    000090 INPUT-OUTPUT SECTION.
  00011    000100 FILE-CONTROL.
  00012    000110     SELECT PERSONNEL-MASTER-FILE
  00013    000120         ASSIGN TO SYS006-UT-2314-S.
  00014    000130     SELECT PRINTER
  00015    000140         ASSIGN TO SYS005-UR-1403-S.
  00016    000150 DATA DIVISION.
  00017    000160 FILE SECTION.
  00018    000170 FD  PERSONNEL-MASTER-FILE
  00019    000180     LABEL RECORDS STANDARD.
  00020    000190 01  IM01-PERSONNEL-MASTER-RCD.
  00021    000200     05  IM01-SOC-SEC-NUM                        PIC X(9).
  00022    000210     05  IM01-NAME                               PIC X(19).
  00023    000220     05  IM01-DEPARTMENT                         PIC X(2).
  00024    000230     05  FILLER                                  PIC X(5).
  00025    000240     05  IM01-NUMBER-DEPENDENTS                  PIC 9(2).
  00026    000250     05  IM01-MONTHLY-PAY                        PIC 9(4)V9(2).
  00027    000260     05  FILLER                                  PIC X(38).
  00028    000270 FD  PRINTER
  00029    000280     LABEL RECORDS OMITTED.
  00030    000290 01  OP01-PRINTER-RCD                            PIC X(133).
  00031    000300 WORKING-STORAGE SECTION.
  00032    000310 77  WS-MASTER-FILE-EOF-SWITCH                   PIC X.
  00033    000320 77  WS-TOTAL-FAMILY-SIZE-SUM                    PIC S9(3)
  00034 I                                                 VALUE IS ZERO
  00035    000330                                         USAGE IS COMP-3.
  00036    000340 77  WS-TOTAL-NET-PAY-SUM                        PIC S9(5)V9(2)
  00037 I                                                 VALUE IS ZERO
  00038    000350                                         USAGE IS COMP-3.
  00039    000360 77  WS-DEPARTMENT-HOLD                          PIC X(2)
  00040    000370                                         VALUE IS SPACES.
  00041    000380 77  WS-FEDERAL-TAX-HOLD                         PIC S9(4)V9(2)
  00042    000390                                         USAGE IS COMP-3.
  00043    000400 77  WS-FICA-TAX-HOLD                            PIC S9(4)V9(2)
  00044    000410                                         USAGE IS COMP-3.
  00045    000420 77  WS-NET-PAY-HOLD                             PIC S9(4)V9(2)
  00046    000430                                         USAGE IS COMP-3.
  00047    000440 77  WS-FAMILY-SIZE-HOLD                         PIC S9(3)
  00048    000450                                         USAGE IS COMP-3.
  00049    000460 77  WS-TAX-RATE-HOLD                            PIC S9V9(2)
  00050    000470                                         USAGE IS COMP-3.
  00051    000480 77  WS-FICA-TAX-RATE                            PIC SV9(3)
  00052 I                                                 VALUE IS +.059
```

```
00053   000490                                          USAGE IS COMP-3.
00054   000500 77  WS-0-DEPENDENTS-TAX-RATE                       PIC SV9(3)
00055   000510                                          USAGE IS COMP-3
00056   000520                                          VALUE IS +.2.
00057   000530 77  WS-1-4-DEPENDENTS-TAX-RATE                     PIC SV9(3)
00058   000540                                          USAGE IS COMP-3
00059   000550                                          VALUE IS +.15.
00060   000560 77  WS-5-8-DEPENDENTS-TAX-RATE                     PIC SV9(3)
00061   000570                                          USAGE IS COMP-3
00062   000580                                          VALUE IS +.12.
00063   000590 77  WS-9-12-DEPENDENTS-TAX-RATE                    PIC SV9(3)
00064   000600                                          USAGE IS COMP-3
00065   000610                                          VALUE IS +.09.
00066   000620 77  WS-13-99-DEPENDENTS-TAX-RATE                   PIC SV9(3)
00067   000630                                          USAGE IS COMP-3
00068   000640                                          VALUE IS +.05.
00069   000650 77  WS-SWITCH-ON                                   PIC X
00070   000660                                          VALUE IS '0'.
00071   000670 77  WS-SWITCH-OFF                                  PIC X
00072   000680                                          VALUE IS '1'.
00073   000690 01  WS-COLUMN-HEADING.
00074   000700     05  FILLER                                     PIC X(14)
00075   000710                                          VALUE IS SPACES.
00076   000720     05  FILLER                                     PIC X(18)
00077   000730                                          VALUE IS 'SSN'.
00078   000740     05  FILLER                                     PIC X(15)
00079   000750                                          VALUE IS 'NAME'.
00080   000760     05  FILLER                                     PIC X(13)
00081   000770                                          VALUE IS 'MONTHLY PAY'.
00082   000780     05  FILLER                                     PIC X(11)
00083   000790                                          VALUE IS 'NO OF DEPS'.
00084 I            05  FILLER                                     PIC X(11)
00085   000810                                          VALUE IS 'PAY AVE'.
00086 I            05  FILLER                                     PIC X(10)
00087   000830                                          VALUE IS 'FED TAX'.
00088 I            05  FILLER                                     PIC X(12)
00089 I                                                 VALUE IS 'TAX RATE'.
00090 I            05  FILLER                                     PIC X(11)
00091 I                                                 VALUE 'FICA TAX'.
00092 I            05 FILLER                                      PIC X(12)
00093 I                                                 VALUE 'NET PAY'.
00094   000880 01  WS-DETAIL-LINE.
00095   000890     05  FILLER                                     PIC X(11)
00096   000900                                          VALUE SPACES.
00097   000910     05  WS-SOC-SEC-NUM                             PIC X(9).
00098   000920     05  FILLER                                     PIC X(5)
00099   000930                                          VALUE IS SPACES.
00100   000940     05  WS-NAME                                    PIC X(18).
00101   000950     05  FILLER                                     PIC X(6)
00102   000960                                          VALUE SPACES.
00103   000970     05  WS-MONTHLY-PAY                             PIC 99999.99.
00104   000980     05  FILLER                                     PIC X(6)
00105   000990                                          VALUE SPACES.
```

```
001000        05  WS-NUMBER-DEPENDENTS              PIC Z9.
001010        05  FILLER                            PIC X(6)
001020                                          VALUE IS SPACES.
001030        05  WS-PAY-AVERAGE                    PIC $$$$9.99.
001040        05  FILLER                            PIC X(2)
001050                                          VALUE IS SPACES.
001060        05  WS-FEDERAL-TAX                    PIC $$$$9.99.
001070        05  FILLER                            PIC X(5)
001080                                          VALUE IS SPACES.
001090        05  WS-TAX-RATE                       PIC VZ9.
001100        05  FILLER                            PIC X(7)
001110                                          VALUE IS ' %'.
001120        05  WS-FICA-TAX                       PIC $$$$9.99.
001130        05  FILLER                            PIC X(3)
001140                                          VALUE IS SPACES.
001150        05  WS-NET-PAY                        PIC $$$$9.99.
001160        05  FILLER                            PIC X(11)
001170                                          VALUE IS SPACES.
001180 01  WS-FOOTING-LINE.
001190        05  FILLER                            PIC X(35)
001200                                          VALUE IS SPACES.
001210        05  FILLER                            PIC X(31)
001220                                          VALUE IS
001230            'DEPARTMENT TOTAL FAMILY SIZE'.
001240        05  WS-TOTAL-FAMILY-SIZE              PIC ZZ9.
001250        05  FILLER                            PIC X(21)
001260                                          VALUE IS ' AND TOTAL NET PAY'.
001270        05  WS-TOTAL-NET-PAY                  PIC $$$$9.99.
001280        05  FILLER                            PIC X(34)
001290                                          VALUE IS SPACES.
001300 PROCEDURE DIVISION.
001310        OPEN INPUT PERSONNEL-MASTER-FILE
001320             OUTPUT PRINTER.
001330        MOVE WS-SWITCH-OFF TO WS-MASTER-FILE-EOF-SWITCH.
001340        PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT
001350          UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.
001360        CLOSE PERSONNEL-MASTER-FILE
001370              PRINTER.
001380        STOP RUN.
001390 0010-READ-AND-PRINT.
001400        READ PERSONNEL-MASTER-FILE
001410          AT END
001420            PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
001430            MOVE WS-SWITCH-ON TO WS-MASTER-FILE-EOF-SWITCH
001440            GO TO 0010-EXIT.
001450        IF WS-DEPARTMENT-HOLD EQUAL TO SPACES
001460          PERFORM 0030-HEADING-RTN THRU 0030-EXIT
001470        ELSE
001480          IF WS-DEPARTMENT-HOLD NOT EQUAL TO IN01-DEPARTMENT
001490            PERFORM 0020-TOTALS-RTN THRU 0020-EXIT
001500            PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
001510        IF IN01-NUMBER-DEPENDENTS EQUAL TO ZEROS
001520          MOVE WS-0-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
```

```
00159   001530      ELSE
00160   001540        IF IM01-NUMBER-DEPENDENTS LESS THAN 5
00161   001550          MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00162   001560        ELSE
00163   001570          IF IM01-NUMBER-DEPENDENTS LESS THAN 9
00164   001580            MOVE WS-5-8-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00165   001590          ELSE
00166   001600            IF IM01-NUMBER-DEPENDENTS LESS THAN 13
00167   001610              MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00168   001620            ELSE
00169   001630              MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
00170   001640                WS-TAX-RATE-HOLD.
00171 I             ADD 1 IM01-NUMBER-DEPENDENTS
00172 I               GIVING WS-FAMILY-SIZE-HOLD.
00173   001650      MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
00174   001660        GIVING WS-FEDERAL-TAX-HOLD.
00175   001670      MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
00176   001680        GIVING WS-FICA-TAX-HOLD.
00177   001690      SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
00178   001700        IM01-MONTHLY-PAY
00179   001710        GIVING WS-NET-PAY-HOLD
00180   001720      DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
00181   001730        GIVING WS-PAY-AVERAGE.
00182   001740      MOVE IM01-SOC-SEC-NUM TO WS-SOC-SEC-NUM.
00183   001750      MOVE IM01-NAME TO WS-NAME.
00184   001760      MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
00185   001770      MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
00186   001780      MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
00187   001790      MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
00188   001800      MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
00189   001810      MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
00190   001820      ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
00191   001830      ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
00192   001840      WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
00193   001850        AFTER ADVANCING 2 LINES.
00194   001860 0010-EXIT. EXIT.
00195   001870 0020-TOTALS-RTN.
00196   001880      MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
00197   001890      MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
00198   001900      WRITE OR01-PRINTER-RCD FROM WS-FOOTING-LINE
00199   001910        AFTER ADVANCING 3 LINES.
00200   001920      MOVE ZEROS TO WS-TOTAL-FAMILY-SIZE-SUM
00201   001930        WS-TOTAL-NET-PAY-SUM.
00202   001940 0020-EXIT. EXIT.
00203   001950 0030-HEADING-RTN.
00204   001960      MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
00205   001970      WRITE OR01-PRINTER-RCD FROM WS-COLUMN-HEADING
00206   001980        AFTER ADVANCING CHAN1.
00207   001990 0030-EXIT. EXIT.
```

5

```
INTRNL NAME     LVL  SOURCE NAME                          BASE    DISPL   INTRNL NAME    DEFINITION    USAGE        R  O  Q  M

DNM=1-151       FD   PERSONNEL-MASTER-FILE                DTF=01          DNM=1-151                    DTFSD                 F
DNM=1-196       01   IM01-PERSONNEL-MASTER-RCD             BL=1    000    DNM=1-196      DS 0CL80      GROUP
DNM=1-234       02   IM01-SOC-SEC-NUM                      BL=1    000    DNM=1-234      DS 9C         DISP
DNM=1-260       02   IM01-NAME                             BL=1    009    DNM=1-260      DS 18C        DISP
DNM=1-279       02   IM01-DEPARTMENT                       BL=1    01B    DNM=1-279      DS 2C         DISP
DNM=1-304       02   FILLER                                BL=1    01D    DNM=1-304      DS 5C         DISP
DNM=1-320       02   IM01-NUMBER-DEPENDENTS                BL=1    022    DNM=1-320      DS 2C         DISP-NM
DNM=1-352       02   IM01-MONTHLY-PAY                      BL=1    024    DNM=1-352      DS 6C         DISP-NM
DNM=1-378       02   FILLER                                BL=1    02A    DNM=1-378      DS 38C        DISP
DNM=1-397       FD   PRINTER                              DTF=02          DNM=1-397                    DTFPR                 F
DNM=1-428       01   OR01-PRINTER-RCD                      BL=2    000    DNM=1-428      DS 133C       DISP
DNM=1-454       77   WS-MASTER-FILE-EOF-SWITCH             BL=3    000    DNM=1-454      DS 1C         DISP
DNM=2-000       77   WS-TOTAL-FAMILY-SIZE-SUM              BL=3    001    DNM=2-000      DS 2P         COMP-3
DNM=2-034       77   WS-TOTAL-NET-PAY-SUM                  BL=3    003    DNM=2-034      DS 4P         COMP-3
DNM=2-067       77   WS-DEPARTMENT-HOLD                    BL=3    007    DNM=2-067      DS 2C         DISP
DNM=2-095       77   WS-FEDERAL-TAX-HOLD                   BL=3    009    DNM=2-095      DS 4P         COMP-3
DNM=2-124       77   WS-FICA-TAX-HOLD                      BL=3    00D    DNM=2-124      DS 4P         COMP-3
DNM=2-150       77   WS-NET-PAY-HOLD                       BL=3    011    DNM=2-150      DS 4P         COMP-3
DNM=2-178       77   WS-FAMILY-SIZE-HOLD                   BL=3    015    DNM=2-178      DS 2P         COMP-3
DNM=2-207       77   WS-TAX-RATE-HOLD                      BL=3    017    DNM=2-207      DS 2P         COMP-3
DNM=2-233       77   WS-FICA-TAX-RATE                      BL=3    019    DNM=2-233      DS 2P         COMP-3
DNM=2-259       77   WS-0-DEPENDENTS-TAX-RATE              BL=3    01B    DNM=2-259      DS 2P         COMP-3
DNM=2-293       77   WS-1-4-DEPENDENTS-TAX-RATE            BL=3    01D    DNM=2-293      DS 2P         COMP-3
DNM=2-329       77   WS-5-8-DEPENDENTS-TAX-RATE            BL=3    01F    DNM=2-329      DS 2P         COMP-3
DNM=2-365       77   WS-9-12-DEPENDENTS-TAX-RATE           BL=3    021    DNM=2-365      DS 2P         COMP-3
DNM=2-402       77   WS-13-99-DEPENDENTS-TAX-RATE          BL=3    023    DNM=2-402      DS 2P         COMP-3
DNM=2-440       77   WS-SWITCH-ON                          BL=3    025    DNM=2-440      DS 1C         DISP
DNM=2-462       77   WS-SWITCH-OFF                         BL=3    026    DNM=2-462      DS 1C         DISP
DNM=3-000       01   WS-COLUMN-HEADING                     BL=3    028    DNM=3-000      DS 0CL127     GROUP
DNM=3-030       02   FILLER                                BL=3    028    DNM=3-030      DS 14C        DISP
DNM=3-049       02   FILLER                                BL=3    036    DNM=3-049      DS 18C        DISP
DNM=3-068       02   FILLER                                BL=3    048    DNM=3-068      DS 15C        DISP
DNM=3-087       02   FILLER                                BL=3    057    DNM=3-087      DS 13C        DISP
DNM=3-106       02   FILLER                                BL=3    064    DNM=3-106      DS 11C        DISP
DNM=3-125       02   FILLER                                BL=3    06F    DNM=3-125      DS 11C        DISP
DNM=3-144       02   FILLER                                BL=3    07A    DNM=3-144      DS 10C        DISP
DNM=3-163       02   FILLER                                BL=3    084    DNM=3-163      DS 12C        DISP
DNM=3-182       02   FILLER                                BL=3    090    DNM=3-182      DS 11C        DISP
DNM=3-201       02   FILLER                                BL=3    09B    DNM=3-201      DS 12C        DISP
DNM=3-220       01   WS-DETAIL-LINE                        BL=3    0A8    DNM=3-220      DS 0CL133     GROUP
DNM=3-250       02   FILLER                                BL=3    0A8    DNM=3-250      DS 11C        DISP
DNM=3-269       02   WS-SOC-SEC-NUM                        BL=3    0B3    DNM=3-269      DS 9C         DISP
DNM=3-293       02   FILLER                                BL=3    0BC    DNM=3-293      DS 5C         DISP
DNM=3-312       02   WS-NAME                               BL=3    0C1    DNM=3-312      DS 18C        DISP
DNM=3-329       02   FILLER                                BL=3    0D3    DNM=3-329      DS 6C         DISP
DNM=3-348       02   WS-MONTHLY-PAY                        BL=3    0D9    DNM=3-348      DS 8C         NM-EDIT
DNM=3-383       02   FILLER                                BL=3    0E1    DNM=3-383      DS 6C         DISP
DNM=3-402       02   WS-NUMBER-DEPENDENTS                  BL=3    0E7    DNM=3-402      DS 2C         NM-EDIT
DNM=3-439       02   FILLER                                BL=3    0E9    DNM=3-439      DS 6C         DISP
DNM=3-458       02   WS-PAY-AVERAGE                        BL=3    0EF    DNM=3-458      DS 8C         NM-EDIT
```

```
                    MEMORY MAP

           TGT                          00549

        SAVE AREA                       0C54B
        SWITCH                          0C55C
        TALLY                           0C594
        SORT SAVE                       0C598
        ENTRY-SAVE                      CC59C
        SORT CORE SIZE                  005A0
        NSTD-REELS                      005A4
        SORT RET                        0C5A6
        WORKING CELLS                   005A8
        SORT FILE SIZE                  0C6D8
        SORT MODE SIZE                  0C6DC
        PGT-VN TBL                      C06FC
        TGT-VN TBL                      CC6E4
        SORTAB ADDRESS                  0C6E8
        LENGTH OF VN TBL                0C6EC
        LNGTH OF SORTAB                 CC6EE
        PGM ID                          006F0
        A(INIT1)                        0C6F8
        UPSI SWITCHES                   0C6FC
        OVERFLOW CELLS                  0C704
        BL CELLS                        00704
        DTFADR CELLS                    0C710
        TEMP STORAGE                    0C718
        TEMP STORAGE-2                  0C728
        TEMP STORAGE-3                  CC738
        TEMP STORAGE-4                  00738
        BLL CELLS                       0C738
        VLC CELLS                       CC73C
        SBL CELLS                       0C73C
        INDEX CELLS                     0C73C
        SUBADR CELLS                    0073C
        ONCTL CELLS                     0C73C
        PFMCTL CELLS                    0073C
        PFMSAV CELLS                    0C73C
        VN CELLS                        C0750
        SAVE AREA =2                    0075C
        XSASW CELLS                     0075C
        XSA CELLS                       0C75C
        PARAM CELLS                     0075C
        RPTSAV AREA                     CC75C
        CHECKPT CTR                     0C75C
        IOPTR CELLS                     0075C

LITERAL POOL (HEX)

007D0 (LIT+0)      CF5C9CC1  3C1C4020  5B202021  204B2020  40202120  F0202120
007E8 (LIT+24)     000CC0C0  0C0C0004  5B5BC2D6  C7C5D540  5B5BC2C3  D3D6E2C5
```

IA-02-02-14

```
                PGT                                    OC768

             CVERFLOW CELLS                            CC768
             VIRTUAL CELLS                             OC768
             PRCCEDIRE NAME CELLS                      OC76C
             GENERATED NAME CELLS                      CG784
             SUPCTF ADDRESS CELLS                      OC7C4
             VNI CELLS                                 OC7C4
             LITERALS                                  007D0
             DISPLAY LITERALS                          CC8C0

 REGISTER ASSIGNMENT

  REG 6    BL =3
  REG 7    BL =1
  REG 8    BL =2

137            CCC800                        START   EQU   *
               OCC80C  58 10 D 1C8                   L     1,1C8(0,13)          DTF=1
               000804  48 10 C 086                   SH    1,086(0,12)          LIT+30
               OC0808  94 8F 1 000                   NI    000(1),X'8F'
               OCC80C  41 10 C 088                   LA    1,088(0,12)          LIT+32
               OC0810  58 00 D 1C8                   L     0,1C8(0,13)          DTF=1
               0C0814  18 40                         LR    4,0
               OCC816  C5 F0                         BALR  15,0
               OCC818  50 00 F CC8                   ST    0,008(0,15)
               00081C  45 00 F 00C                   BAL   0,00C(0,15)
               OC082C  COC00000                      DC    X'00000000'
               OCC824  0A 02                         SVC   2
               OCC826  41 10 C 088                   LA    1,088(0,12)          LIT+32
               00082A  58 00 D 1CC                   L     C,1CC(0,13)          DTF=2
               OCC32E  18 40                         LR    4,0
               000830  C7 00                         BCR   0,0
               OCC832  05 F0                         BALR  15,0
               000834  50 00 F CC8                   ST    0,008(0,15)
               000838  45 00 F 00C                   BAL   0,00C(0,15)
               OCC83C  COC0C000                      DC    X'00000000'
               OCC840  0A 02                         SVC   2
               000842  50 20 C 1C0                   ST    2,1C0(0,13)          BL =2
               OCC846  58 80 C 1C0                   L     8,1C0(0,13)          BL =2
139            OCC84A  D2 00 6 CC0 6 026             MVC   000(1,6),026(6)      DNM=1-454      DNM=2-462
140            000850  58 G0 D 208                   L     0,208(0,13)          VN=01
               0C0854  5C 00 D 1F4                   ST    0,1F4(0,13)          PSV=1
               0CC858  58 00 C 01C                   L     0,01C(0,12)          GN=01
               OCC85C  50 00 D 208                   ST    0,208(0,13)          VN=01
               00C86C                        GN=01   EQU   *
               000860  58 20 C 020                   L     2,020(0,12)          GN=02
               OCC864  D5 00 6 C00 6 025             CLC   000(1,6),025(6)      DNM=1-454      DNM=2-440
               3CC86A  07 82                         BCR   8,2
               OCC86C  58 10 C 004                   L     1,004(0,12)          PN=01
```

```
          00091A   58 00 D 1F8                        L      0,1F8(0,13)          PSV=2
          00091E   50 00 D 20C                        ST     0,20C(0,13)          VN=02
149       000922   D2 0C 6 000 6 025                  MVC    000(1,6),025(6)      DNM=1-454        DNM=2-440
150       000928   58 10 C 008                        L      1,008(0,12)          PN=02
          00092C   07 F1                              BCR    15,1
151       00092E                           GN=04      EQU    *
          00092E   58 20 C 030                        L      2,030(0,12)          GN=06
          000932   95 40 6 007                        CLI    007(6),X'40'         DNM=2-67
          000936   07 72                              BCR    7,2
          000938   D5 00 6 008 6 007                  CLC    008(1,6),007(6)      DNM=2-67+1       DNM=2-67
          00093E   07 72                              BCR    7,2
152       000940   58 00 D 210                        L      0,210(0,13)          VN=03
          000944   50 00 D 1FC                        ST     0,1FC(0,13)          PSV=3
          000948   58 00 C 034                        L      0,034(0,12)          GN=07
          00094C   50 00 D 210                        ST     0,210(0,13)          VN=03
          000950   58 10 C 010                        L      1,010(0,12)          PN=04
          000954   07 F1                              BCR    15,1
          000956                           GN=07      EQU    *
          000956   58 00 D 1FC                        L      0,1FC(0,13)          PSV=3
          00095A   50 00 D 210                        ST     0,210(0,13)          VN=03
154       00095E   58 10 C 038                        L      1,038(0,12)          GN=08
          000962   07 F1                              BCR    15,1
154       000964                           GN=06      EQU    *
          000964   58 20 C 038                        L      2,038(0,12)          GN=08
          000968   D5 01 6 007 7 018                  CLC    007(2,6),018(7)      DNM=2-67         DNM=1-279
          00096E   07 82                              BCR    8,2
155       000970   58 00 D 20C                        L      0,20C(0,13)          VN=02
          000974   50 00 D 200                        ST     0,200(0,13)          PSV=4
          000978   58 00 C 03C                        L      0,03C(0,12)          GN=09
          00097C   50 00 D 20C                        ST     0,20C(0,13)          VN=02
          000980   58 10 C 00C                        L      1,00C(0,12)          PN=03
          000984   07 F1                              BCR    15,1
          000986                           GN=09      EQU    *
          000986   58 00 D 200                        L      0,200(0,13)          PSV=4
          00098A   50 00 D 20C                        ST     0,20C(0,13)          VN=02
156       00098E   58 00 D 210                        L      0,210(0,13)          VN=03
          000992   50 00 D 204                        ST     0,204(0,13)          PSV=5
          000996   58 00 C 040                        L      0,040(0,12)          GN=010
          00099A   50 00 D 210                        ST     0,210(0,13)          VN=03
          00099E   58 10 C 010                        L      1,010(0,12)          PN=04
          0009A2   07 F1                              BCR    15,1
          0009A4                           GN=010     EQU    *
          0009A4   58 00 D 204                        L      0,204(0,13)          PSV=5
          0009A8   50 00 D 210                        ST     0,210(0,13)          VN=03
157       0009AC                           GN=08      EQU    *
          0009AC   F2 71 D 100 7 022                  PACK   100(8,13),022(2,7)   TS=01            DNM=1-320
          0009B2   F9 10 D 106 C 068                  CP     106(2,13),068(1,12)  TS=07            LIT+0
          0009B8   58 F0 C 044                        L      15,044(0,12)         GN=011
          0009BC   07 7F                              BCR    7,15
158       0009BE   F8 71 D 100 6 018                  ZAP    100(8,13),018(2,6)   TS=01            DNM=2-259
          0009C4   F1 76 D 100 D 100                  MVO    100(8,13),100(7,13)  TS=01            TS=01
          0009CA   F8 11 6 017 D 106                  ZAP    017(2,6),106(2,13)   DNM=2-207        TS=07
160       0009D0   58 10 C 048                        L      1,048(0,12)          GN=012
```

```
                                        CROSS-REFERENCE DICTIONARY

DATA NAMES                          DEFN       REFERENCE

PERSONNEL-MASTER-FILE               00012      00137  00137  00142  00146  00146
IM01-SOC-SEC-NUM                    00021      00182
IM01-NAME                           00022      00183
IM01-DEPARTMENT                     00023      00154  00204
IM01-NUMBER-DEPENDENTS              00025      00157  00160  00163  00166  00171  00185
IM01-MONTHLY-PAY                    00026      00173  00175  00177  00184
PRINTER                             00014      00137  00137  00142  00192  00198  00205
OR01-PRINTER-RCD                    00030      00192  00192  00192  00198  00198  00198  00198  00205  00205  00205  00205
                                               00205  00205
WS-MASTER-FILE-EOF-SWITCH           00032      00139  00140  00149
WS-TOTAL-FAMILY-SIZE-SUM            00033      00191  00196  00200
WS-TOTAL-NET-PAY-SUM                00036      00190  00197  00200
WS-DEPARTMENT-HOLD                  00039      00151  00151  00151  00154  00204
WS-FEDERAL-TAX-HOLD                 00041      00173  00173  00177  00186
WS-FICA-TAX-HOLD                    00043      00175  00177  00188
WS-NET-PAY-HOLD                     00045      00177  00177  00180  00189  00190
WS-FAMILY-SIZE-HOLD                 00047      00171  00180  00191
WS-TAX-RATE-HOLD                    00049      00158  00161  00164  00167  00169  00173  00187
WS-FICA-TAX-RATE                    00051      00175
WS-0-DEPENDENTS-TAX-RATE            00054      00158
WS-1-4-DEPENDENTS-TAX-RATE          00057      00161
WS-5-8-DEPENDENTS-TAX-RATE          00060      00164
WS-9-12-DEPENDENTS-TAX-RATE         00063      00167
WS-13-99-DEPENDENTS-TAX-RATE        00066      00169
WS-SWITCH-ON                        00069      00140  00149
WS-SWITCH-OFF                       00071      00139
WS-COLUMN-HEADING                   00073      00205  00205
WS-DETAIL-LINE                      00094      00192  00192
WS-SOC-SEC-NUM                      00097      00182
WS-NAME                             00100      00183
WS-MONTHLY-PAY                      00103      00184
WS-NUMBER-DEPENDENTS                00106      00185
WS-PAY-AVERAGE                      00109      00180
WS-FEDERAL-TAX                      00112      00186
WS-TAX-RATE                         00115      00187
WS-FICA-TAX                         00118      00188
WS-NET-PAY                          00121      00189
WS-FOOTING-LINE                     00124      00198  00199
WS-TOTAL-FAMILY-SIZE                00130      00196
WS-TOTAL-NET-PAY                    00133      00197

PROCEDURE NAMES                     DEFN       REFERENCE

0010-READ-AND-PRINT                 00145      00140
0010-EXIT                           00194      00140  00150
0020-TOTALS-RTN                     00195      00148  00155
0020-EXIT                           00202      00148  00155
```

```
28/08/78   PHASE   XFR-AD  LOCORE  HICORE  DSK-AD   ESD TYPE   LABEL      LOADED  REL-FR

      PHASE***   005000  005000  006723  2E 12 4  CSECT      DUMPOPE    005000  005000

                                                  CSECT      IJGFISWZ   005E38  005E38
                                                  *  ENTRY   IJGFIZWZ   005E38
                                                  *  ENTRY   IJGFIZZZ   005E38
                                                  *  ENTRY   IJGFIEZZ   005E38

                                                  CSECT      ILBDSAE0   0060E0  0060E0
                                                     ENTRY   ILBDSAE1   006100

                                                  CSECT      IJDFAPIZ   005D20  005D20
                                                  *  ENTRY   IJDFAZIZ   005D20

                                                  CSECT      ILBDMNS0   006008  006008

* UNREFERENCED SYMBOLS                            WXTRN      STXITPSW
                                                  WXTRN      ILBDDBG2

002 UNRESOLVED ADDRESS CONSTANTS
```

```
   SSN            NAME              MONTHLY PAY  NO OF DEPS PAY AVE     FED TAX   TAX RATE     FICA TAX    NET PAY

784764025   GRUNDY GARVEY JR          $568.00       12         $37.66    $45.44     08 %        $32.94     $489.62

639167039   MOTT ROY                  $398.00        9         $34.30    $31.84     08 %        $23.08     $343.08

257808345   CHANEY HELEN WILLI        $344.00        8         $31.41    $41.28     12 %        $19.95     $282.77

                      DEPARTMENT TOTAL FAMILY SIZE    32 AND TOTAL NET PAY   $1115.47
   SSN            NAME              MONTHLY PAY  NO OF DEPS PAY AVE     FED TAX   TAX RATE     FICA TAX    NET PAY

535411804   RITCHISON BENNIE L        $568.00        0        $421.46   $113.60     20 %        $32.94     $421.46

                      DEPARTMENT TOTAL FAMILY SIZE     1 AND TOTAL NET PAY    $421.46
```

IA-02-02-14

```
       SSN                  NAME              MONTHLY PAY  NO OF DEPS PAY AVE     FED TAX   TAX RATE    FICA TAX   NET PAY




0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 0C5982 - CONDITION CODE 3 - DATA EXCEPTION
0S00I JOB DUPPPE3  CANCELED
```

```
         DUMPPE3            28/08/78                                                                       22.49.06                      PAGE   1

GR 0-7   00005C54 000059A4 000054C0 0CCC5000   CC0052R8 50005CE0 0000505R 000053D8
GR 8-F   CC0C54CC CCCC5C80 0C009000 CCC05000   CCJC5768 0000554R 80005C96 00005020
COMREG   PC ADDR IS C00368

000C00   003C0000 GCC0000J 00004C6A C0000438   CC0C0010 00000368 FF050000 00000000    ................  ................
000C20   FF050300 40C04888 FF150007 F0C05398   5859C2C5 0601F3C1 FF350130 80035E5A    .... .......0...  $$9EOJ3A........
0CCC4C   CCCC185C CCC0C0C0 CG0C182P CCCCC000   F6205500 026E29FC 0C040000 0F0023D2    ...&............  6..............K
00006C   0C04CC0G CC0376F4 CC04C00C CC002392   CCCCC00C 000011E4 0C040000 0003F6A4    ........4.......  ........U.......
0C0CPC   58F2F6F2 C6C2C8F6 CCCCC0CG CCC0C003   CC05C003 000306PC 06800650 41880073    $762FBH6........  ................
0000A0   457C0148 540FC393 41A00658 457081P0.  18A84190 01784180 C1A047F0 000C0668    .......C...0....  .........A..0...
0000C0   D6RCC6P0 C6P006R0 06R0C680 41P80017   41P80C50 4570014P 41800178 9640A001    ................  ...&.........; ..
0000E0   51204C0C 471000FC 92604001 95E2A002   4780848A 95C1A002 4780849A 9561A002    ..........-...S..  .....A......../..
000100   47800110 51C4A000 478CC11C 920300AF   9281A00C 48A003C2 48740604 49A003D6    ................  ........B..O....O
0C0120   41AAE64P C7787459 7038E701 7C587058   9283A000 9680A0C1 4400095A 0788947F    ..n....9..P.....  ................
00014C   A001457C CC9A07F8 4280C0F5 588GC2P8   5190A000 07175890 A0044890 02225000    ........B....Z....  ...............&.
0C0160   902C021P 90105298 02C79008 8CCC96P0   AC0C07F7 45700150 D2078A32 8A3858E0    ..K......K.......  ...7...&K........
0C01RC   C2C4982C F77A4780 018F5285 C6A8CCC7   FA32C648 1PAAD008 BA32000C 43A10009    .D...........O...  ..D..............
0C01A0   42800397 41AAD648 D2CCC589 03979540   C5894740 01CC9560 05894780 01C69240    .......O.K......   ... ...-......F.
0C01CC   C5RF47F0 C1CC0200 C58SC588 4590020A   47F0C662 5890A004 4220A000 9140A001    ...O...K.........  .O............ ..
0C01EC   471CCRC4 C2070294 9C084PA0 03C241AA   C6C4D201 0016A000 95990010 82000295    ...FK..........B..  O.K.............
0C0200   459C02CA 4400A004 5890A004 98C8902C   91020021 4780021C 9200059A 989F0298    ................  ................
0C0220   P70C0020 5264D6A3 91E0877A 47800262   9120877A 47100254 91908R6A 4710C254    .......O........  ................
0C0240   91038?7P 47700179 9503C093 4780C178   47F00262 0207C258 8ACC9890 8AC88200    ................  .O..K.........H..
0C026C   02590257 025PRA98 989CPAA0 P2000298   968JA000 41100030 47F08F9E 96030039    ..K.............  ..........O......
00C280   56010C38 P7000038 00615C40 41504044   FCCC0C0C 00000000 FF050133 80004884    .........../* /& .  O...............
0C02A0   00C38FA8 50038094 00037D89 0303FAF0   CC034CC4 80001DAE 00001000 00002000    ...............O  ...D............
0002C0   0CCC3CCC CCC04000 36C2C500 C5P905PA   C7599068 05A09879 00505970 8A2447D0    ....... ..KN.....  ..........&......
0C02E0   02EARA7C CC0847F0 02EE8P70 CCC81F78   1P871C78 5E7005RC 50900590 9130059A    ........O........  ..........&.......
0C030C   477C084C 12774740 07401866 4A7005R0   5C600584 406005R0 584005C0 19675E60    ....... . ......  .-.. -..........-
00032C   RG2C5C60 PC205PR0 05CP1R67 5F6GRC20   5C6CP020 588005C4 5E70R020 50708020    ..&-.....H...-..  &-.....D....&...
0C0340   D20005RA 05899R68 05ACC7F9 19664360   05RAP860 00025P86 05C05E70 801C5070    K............9...-  ...-............&.
0C036C   801C47FC 03400000 F2F861F0 F861F7F8   5CCC5010 0000000C 00000000 00000000    ...O. ..28/08/78  &.&.............
0C0380   C4F4D4C7 C7C5F340 0001C7FF 0CC06223   CCC06223 000000C0 0003FFFF DAF04E80    DUMPPE3 ..P......  ...............*.
0003A0   2PAC7EC0 CC6337C9 370C3991 3992345C   3C703C74 3C7838F2 F8F0F8F7 F8F2F4F0    ........I.........*  ........280878240
0003CC   C0003648 CCC0U014 1P60159A 1A3C1A4C   1A5CC020 378C0010 5858C206 00060007    ..........-......  .*.......$$BO....
0C03F0   010C1P04 36040000 0000C000 03681080   C00003F4 00000000 00003E7C 00000044    ................  ...4............
00040C   00001A74 CCC0CC00 000CC000 CCCCC000   CCCC0000 00000000 00000000 00000000    ................  ................
0C042C   0C0CC00C CC004500 CC0CC5A0 00003F38   CCCC0C0C 00CC0000 92380223 909FC298    ................  ................
0C0440   58PE0288 47F0061C 47F00646 4AA003C2   9589A000 47800468 9190A000 4710047A    .....O....O.....B  ................
0C046C   58R0AC04 9CC8802C 58PEC2P8 5C69C5A0   459C04E0 98690540 07F99601 A00099RE    ................  ...........9......
0C0480   02889C69 C5A04590 04F09869 05A095FF   A00F0789 91A0977A 478004C2 9640877A    ................  ..............B. ..
0C04AC   547F877A C218RAC8 02985000 PAE44PE0   C222D207 RAC0E000 94FDR4C1 95RE02RR    ....K..H..&..U..  ..K..........A....
0C04C0   07F95000 RAPC4PE0 0222C207 RA59FCCC   5PE002C4 94FD8459 D2198AA0 029807F9    .9&........K.....  ...D....K.......9
0004EC   D2CC0989 05PA9879 0C5C5970 PA2447D0   04FARA70 000847F0 04FE8870 00081F78    K..........&......  ........O.........
0C0500   1P971C78 5E7005RC 9180A000 4780051E   55FFAC0F 4780051E D2000598 05DA9130    ................  .........K.......
0C0520   05RA4770 C5729140 05RAC719 5080059C   12774740 056C1R66 4A700592 5D60C5R4    ........ ....&...  ... ..........-..
000540   40600582 58P005CC 1F675F6C 80245C60   8C245PR0 05C81867 5E60R024 50608024     -..........-..&-  .....H...-..&-..
0C056C   58R0C5C4 5E7GR074 507C8074 9240C58A   C7F99529 02230789 5090059C 18664360    ...D.....&..... ..  .9........&.......-
0C0580   05RA5F6C CC025PR6 05C05E7C 8C1C5C70   8C1C552C 02230789 9240059A C7F90000    ...-...........&.  ........... ...9..
0C05A0   000021FF 9000C178 0CC0C020 800005E4   CC0C0000 00000001 10101000 FE87F598    ...............U  ...............S.
0005C0   C000425R CC003F39 0CCC403C CCC04128   CCCC0000 00000050 909F0298 92200223    ........... .....  .........&.........
0005F0   459C04C8 41900200 5500C023 478CC8C2   5535CC23 478006C8 48600022 1A664870    ...H...........K  ...........-......
CC0600   02C84EA6 7C0007F6 F2E5C3F1 F0F09564   00234770 00C09200 002307F1 41900080    .H......6SVC100..  ..............1....
```

```
0043AC  0003A0AC CCC00030 0003FFFF CAFD4580    20004E80 006337C9 37003991 34923A5C    ...............+.  ..+....1.......*
0C43C0  3C703C74 2C7838F2 F8F0F8F7 F8F2F4F0    C0003648 00000014 19201996 1A3C144C    .......280878240  ................
0C43EC  1A5CCC2C 378C0C3C CCCCC000 000CCCC0    CC000000 3604000C 00000000 03680080    .*..............  ................
004400  00004444 CCC00000 00003E7C CC000344    CCC01A74 00000000 00000000 00000000    ................  ................
0C442C  CCCCCCCC CCC0CC0C CC00C000 CCCC0CC0    CCC00C0C 0000000C 00000540 0C004030    ................  ............... .
004440  0CCCC00C CCC00000 00003E7C 00C00044    CC001A74 00000000 00000000 00000000    ................  ................
004460  CC000000 00000000 00000000 CCCCCC00    CCCC0C00 000000C0 00000540 00004128    ................  ................
0044EC  CCCCCCCC --SAME--                                                              ....
004500  C3C9F0C0 CC1C000F C3D7F000 001D000F    C7D9FC00 E01E000F C4D2F013 0130130F    CR0......CP0.....  PR0.....DK0.....
0C4520  C4D2F113 1131131F C4D2F213 2132132F    C4D2F313 3133133F C4D2F413 4134134F    DK1......DK2.....  DK3.....DK4.....
0C4540  C4D2F514 014014CF C4D2F614 1141141F    C4C2F714 2142142F E3D7F018 0180180F    DK5.. ..DK6.....  DK7.....TP0.....
0C4560  E3D7F118 1181181F E3D7F218 2182182F    E3D7F318 3183183F E3D7F418 4184194F    TP1.....TP2.....  TP3.....TP4.....
004580  F3D7F518 5185185F E3C4F018 C180180F    C5D5C440 40000000 C5D5C440 40004500    TP5.....TD0.....  END ...END ...
0C45AC  CCC00000 --SAME--                                                              ....
004760  5B53C2C4 E4D4D7C2 5B5BC2C4 E4C4C7C2    CCC00001 00000006 C6040308 4C404040    $$BDUMPB$$BCUMPB  ...........0
0047AC  40404C40 --SAME--
0048CC  404C4040 40404040 40404040 40404040

LALTYP  HEX LENGTH IS C00C
--8G--
004FA0                    D5D640D5 C1D4C540    FF150007 F00359B8 00005C90 00005000               NO NAME  ....0.....*...&.
004FCC  CC0C50CC CC005768 CC005548 8C0C5C8A    CCCC5D2C 00005C94 00005944 000054C0    ..&............*.  ......*.........
004FEC  0CC05000 000052B8 50005CE0 000C50E8    C0005308 000054C0 00000000 01775529    ..&.....&.*...&Y  ...0............
005000  05FCC70C 500EF00A 47F0F082 0CCC50C0    CCCC5C00 0001D7FF 00005000 00010784    .0.....0..00...&.  ..&...P...&...P.
0C5C20  FFFFFF7C 0C005000 00004F88 CCC06570    CAC4C7F1 0000501C 0C005010 00005FA8    ......&.........  ...1..&...&.....
00504C  00006FA8 CC000368 D202733E 7299F912    7341733F 47407038 FA107348 73CFF911    ........K......9.  ..... ........9.
005060  73467348 47807014 D2117168 72760202    7187728A D2087192 726D0202 71A37299    .........K.....K.  ....K.....K.....
005080  D287728A 58C0F0C6 58E0C00C 58D0F0CA    95C0E000 4770F0A2 9610DC48 92FFE000    K......0F.......0.  .......0........
0050A0  47FCF0AC 58CEF03A 9CECC00C 185D589F    FC8A9110 00480719 07FF0700 00005C80    .00...0.........  0..............*.
0050C0  00005000 00C05000 00005768 0C005548    CC005800 00005C96 C3D6C2C6 F0F0F0F1    ..&...&.........  ......*.COBF0001
0C50EC  C4E4D4D7 F0D7C540 F1CCCC00 CCCCCCF0    F3001136 0C000329 4C004214 6C001C02    DUMP0PE 1.......0  3...............
0C5100  0C058C20 CC150C12 0CC8CCC5 0CF0F1F0    4C404040 40404040 40404040 4040E2E2    ..............010                55
0C5120  D54C4C40 4040404C 40404040 40404040    C5C1D4C5 40404040 40404040 404040D4    N                 NAME            M
005140  D6D5E3C8 D3E84007 C1E84040 D5D640D6    C64CC4C5 D7E24007 C1E840C1 E5C54040    ONTHLY PAY  NO O  F DEPS PAY AVE
005160  4040C6C5 C440E3C1 E7404040 E3C1E740    D9C1E3C5 40404040 C6C9C3C1 40E3C1E7      FED TAX   TAX    RATE    FICA TAX
0051EC  4C4040D5 C5E34CD7 C1E8404C 404040C6    4C404040 40404040 404040F5 F3F5F4F1       NET PAY    F                 53541
0051A0  F1F8F0F4 4C404040 40D9C9E3 C3C8C9E2    D8D540C2 C5D5D5C9 C540D340 40404040    1804     RITCHIS  ON BENNIE L
0051C0  40405BF5 F6F84BF0 F0404040 40404040    FC404040 4040404C 5BF4F2F1 4BF4F640      $568.00          0        $421.46
0051E0  40405BF1 F1F34BF6 F0404040 404CF2F0    40CC4C40 4040404C 405BF3F2 4BF9F440      $113.60     20    .        $32.94
00520C  4040405B F4F2F14B F4F64040 40404C40    40404040 404040C8 40404040 40404040       $421.46             H
0C522C  40404C4C 40404040 40404040 4C4C4C40    40404040 40404040 404040C4 C5D7C1D9                              DEPAR
005240  F3C4C5D5 E340E3D6 E3C1D340 C6C1D4C9    D3E84CF2 C9E9C540 40404040 F140C1D5    TMENT TOTAL FAMI  LY SIZE      1 AN
0C5260  C44CE3D6 E3C1D340 D5C5E340 D7C1E840    4040405B F4F2F14B F4F64040 40404040    D TOTAL NET PAY       $421.46
005280  404C4040 --SAME--
0C52AC  C0000C50 C0000000 G00C0000 0CCG0000    CCCC000C 00000000 00508204 0D400106    ...&.............  .........&... ..
0052CC  C00C532C C0C05340 04005F38 200EF2E8    E2F0F0F6 40010000 00000000 00078020    ....... ......SY  S006 ...........
0052E0  00CC000C C40053D8 80CCC000 CCCC0C46    CCCCC0C0 00460000 06005904 00C80013    .......Q.........  ..............H..
005300  288CCC4F CC460000 01CC6100 5821CC58    CC0053D8 00000050 0C005427 8E0060E0    ........../......  ...0...&......-.
005320  C70C52F2 4C00C0C6 310052F4 4CCCCC05    C8005328 0000000C 06005380 00000050    ...2 ......4 ...  ...............&
005340  28000C4F CC000000 CC0C8000 0CC00105    C0005370 00005378 00005D20 0A84A8F1    ................  ...............1
0C5360  010054C1 CCC00000 C70C4120 FCC0000C    01005431 20000084 06005418 00000050    ...A............  ...............&
0C538C  F5F3F5F4 F1F1F8F0 F4D9C9E3 C3C8C9E2    C6D540C2 C5D5D5C9 C540D3F0 F2F0F2E2    535411804RITCHIS  ON BENNIE L0202S
0C53A0  C6C3F0F0 FCF5F6F8 F0FC4040 404C4040    40404040 40404040 40404040 40404040    FC00056800
0053C0  4C404040 40404040 404C404C 4C404040    CC1C58E0 F06407FE F8F4F1F6 F2F4F4F4                      ....0...84162444
```

```
        DUMPPE1             28/08/78                                                                              PAGE  12
005 3E0   F4E6C9D5 D2D3C540 D9C9D740 E5C1D540    4C4C40F0 F3F0F2E2 E2C75CF4 F0F4F9F0    4WINKLE RIP VAN      0302SSG*40490
005400   F0F04040 40404040 40404040 40404040    40404040 40404040 40404040 40404040    00
005420   40404040 40404040 E0304770 F05A47F0    F1404040 40404040 40404040 4040E2E2            ....0..0  1               SS
005440   D5404040 40404040 40404040 40404040    D5C1D4C5 40404040 40404040 404040D4    N                NAME           M
005460   D6D5E3C8 D3E840D7 C1E84040 D5D640D6    C640C4C5 D7E240D7 C1E840C1 E5C54040    ONTHLY PAY  NO O  F DEPS PAY AVE
005480   4040C6C5 C440E3C1 E7404040 E3C1E740    D9C1E3C5 40404040 C6C9C3C1 40E3C1E7      FED TAX   TAX   RATE    FICA TAX
0054A0   404040D5 C5E340D7 C1E84040 40404040    40404040 40000000 47F0F04C 0A320000       NET PAY                ....00.....
0054C0   60404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040    -
0054E0   404040C4 C5D7C1D9 E3D4C5D5 E340E3D6    E3C1D340 C6C1D4C9 D3E840E2 C9E9C540       DEPARTMENT TO  TAL FAMILY SIZE
005500   40404040 F140C1D5 C440E3D6 E3C1D340    D5C5E340 D7C1E840 4040405B F4F2F14B        1 AND TOTAL   NET PAY   $421.
005520   F4F64040 40404040 40404040 40404040    40404040 40404040 40404040 40404040    46
005540   4040404C 4C0750E0 00A458F0 1C209101    10044780 F04C9140 10024710 F0465BE0       .&...........  ....0.. ....0...
005560   F06447F0 FC1A58E0 101CC7FE D501F060    FC004770 F05A47FC F0465BE0 F06407FE    0..00........N.0-  ....0..00...0...
005580   615C0000 00000003 00F0F04C 0A320000    3C000C4B 00000000 00000000 000C5B00    /*.......00......  ................
0055A0   00000000 00000000 F024CA07 90CEFCA0    5BE01C16 48C0102E 06C01870 44C0F06C    .........0......0-  ...............0.
0055C0   5ACEF06C F2001028 1017C200 10171016    C0000C48 00000000 00000000 00000000    ..0-K......K......  ................
0055E0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000    ................  ................
005600   00000000 --SAME--                                                           ....
0056E0   00000000 00000000 00000000 00000000    00000000 00000000 00005000 00000000    ................  ..........&.....
005700   00000000 000053D8 00005400 000050E8    0000529A 00005348 00000000 00000C4F    ........Q.......&Y  ................
005720   00000000 0042144C 4C40405B F4F2F14B    F4F60000 00000000 00000000 00005BEE    ........  $421.  46.............$.
005740   00005C00 00005C94 00005C60 00005C94    00005B60 00005C60 00005C94 000000E8    ..&...*...*-..*.  ...-..*-..*....Y
005760   00030308 C0C0C35B 0000600B 00005BD4    00005BE8 00005BFE 00005C60 00005C8E    ............-Q...M  ..$Y..$...*-..*.
005780   00005C94 00005B60 00005872 00005904    0000592E 0000591A 00005964 00005956    ..*....-..........  ................
0057A0   000059AC 00005986 000059A4 000059D6    00005A66 00005A00 00005A2A 00005A54    ...............0  ................
0057C0   00005C4E 00005BEE 00005C6C 00005C94    CF5C9C01 3C1C4020 58202021 2C482020    ..**..$...*-..*.  .*.... .$.......
0057E0   40202120 FC202120 00000000 00000004    5B5BC2D6 D7C5D540 5B5BC2C3 D3D6E2C5     ...0...........  $$BOPEN $$BCLOSE
005800   5810D1C8 4B10C0B6 94BF1000 4110C0A8    5800D1C8 184005FC 5C00F008 4500F00C    ..JH............  ..JH. .0&.0...0.
005820   00005298 CAC24110 C0A858C0 D1CC1840    C7C005F0 5000F008 4500F00C 00005348    ..............J..  ...0&.0...0.....
005840   0A075020 D1C058B0 D1C0D200 AC006026    5800D208 500001F4 5800C01C 5000D208    ..&.J...J.K.-.-.  ..K.&.J4....&.K.
005860   58200070 D5C06CC0 6025C7B2 58100004    C7F15800 D1F4500C D2085830 D1C81840    ....N.-.-.......  .1..J4&.K...JH.
005880   4110C090 C70005FC 5C00F0C8 45C0F00C    C000C0C0 0A025810 D1C84A10 C08605F0    ........0&.0...0.  .........JH.....0
0058A0   91801000 4780F016 41101004 412010E0    C2DF1000 20005800 D1CC1840 4110C090    .......0.........  K........J.. ....
0058C0   C70005F0 5CC0F0C8 45C0F00C CCCCCC0C    CAC20A0E 5810D1C8 58F0C024 91201010    ...0&.0...0......  ......JH.0.......
0058E0   C71F1841 41F0C024 D2021041 FC0158F0    101C4550 F00B502C D1BC5870 D1BC58F0    ......0..K...0..0  ....0.&.J...J..0
005900   C02807FF 58C0D20C 5C00C1F8 5800C02C    5C00020C 5810C00C 07F15830 D1F85000    .......K.&.J8....  &.K.......1..J8&.
005920   C20C02C0 6C00A025 5810C0C8 07F15870    CC3C9540 60070772 D5006009 60070772    K.K.-.-.......1..  ... -...N.-.-...
005940   58CCC21C 50C0D1FC 5800C034 5C00C210    5810C010 07F158C0 D1FC5000 D21C5810    ..K.&.J......&.K.  .....1..J.&.K...
005960   C03807F1 58720C3A D50160C7 7018C782    58C0D20C 5000D200 5800C03C 5000020C    ...1....N.-.....  ..K.&.K......&.K.
005980   5810C00C C7F15800 D20C5000 D20C5800    D2105000 D2045800 C0405000 D2105810    .....1..K.&.K...  K.&.K.... &.K...
0059A0   C01007F1 98C0D204 5000C210 F271D100    7C22F910 D1D6C06B 58F0C044 077FF871    ...1..K.&.K.2.J.  ..9.JO...0.....8.
0059C0   D1D0A01A F176D100 D100F811 6017D106    5810C04A 07F1F271 D1007022 F910D106    J.-.1.J.J.8.-.JO  .....12.J...9.JO
0059E0   C06558F0 CC4C07AF F871C1C0 6010F176    C1D0D100 F8116017 D1065810 C04807F1    ...0.....8.J.-.1.  J.J.8.-.JO......1
005A00   F271D100 7022F910 D1D6C06A 58FCCC50    C7AFF871 D1D0601F F176D100 D100F811    2.J...9.JO...0.&  ..8.J.-.1.J.J.8.
005A20   6017D1D6 5810CC4A 07F1F271 D1007022    F911D1D6 C0A858F0 C05407AF F871D100    -.JO......12.J...  9.JO...0.....8.J.
005A40   6021F176 C100D100 F8116017 D1D65810    CC48C7F1 F871D100 6023F176 D1D0D1D0    -.1.J.J.8.-.JO..  ...18.J.-.1.J.J.
005A60   F8116C17 D1D6F271 D1C07022 FA10C1D6    C06DF911 6015D1D6 F275D170 7024FC51    8.-.JO2.J......JO  ..8.-.JO2.J.....
005A80   D1D76017 F176D1D0 D1D0F176 D1D0D1D0    F8336009 D104940F 6009F275 D1D07024    JK-.1.J.J.1.J.J.  8.-.JM..-.2.J...
005AA0   FC51D1D2 6C19F175 C1D0F1D0 F833600D    C1D4F873 D1D0600D FA33D104 6009F275    ..JK-.1.J.J.8.-.  JM8.J.-...JM-.2.
005AC0   D1D87024 F833D10C D1D4F833 6011D1CC    94CF6011 F873D108 6011FD51 D1DA6015    JO....J.JM8.-.J.  ..-.8.JO-...J.-.
005AE0   F873D10A C1FAD209 D1F0C06E 4110C1E6    FFC9D150 D1DC0F10 92581000 D20760EF    8.JOJ.K.J......JW  ..J.J.....8..K.-.
005B00   D1F2D2CA 6C837000 D21160C1 7009F275    C1097024 D209D1E0 C06E4110 D1E6DF09    JSK.-...K.-A..2.  JO..K.J......JW..
005B20   C1E0D1DC C610925B 1C0CC2C7 AC0901E2    F271C1DA 7022DF03 D1E0C07A CE03D1E0    J.J.....8..K.-RJS  2.JO..K.J......J.
005B40   C1DE0201 AC6FD1E2 E209C1E0 C06E4110    C1E6CF09 D1E06009 0610925B 1000D207    J.K.-XJSK.J.....  JW..J.-.....8..K.
```

```
       DUMPPE3              28/06/78                                                                              PAGE  13

005B60  60F9D1E2 F871D1D8 6017C705 D1D8D1D8   940FD1DE D203D1E0 C07CDE03 D1E0D1DE   -9JS8.JQ-.P.JQJQ  ..J.K.J.....J.J.
005B80  D2016106 D1E2D209 D1E0C06E 4110D1E6   DF05D1E0 60000410 925B1000 D207610F   K./.JSK.J.....JW  ..J.-....$..K./.
005BA0  D1E2D209 D1E0C06E 4110D1E6 DF09D1E0   6011C610 925B1000 D207611A D1E2FA33   JSK.J.....JW..J.  -....$..K./.JS..
005BC0  60036011 FA116001 6015D284 58CC60A8   5810D1CC 92F09000 58F01010 4550F00C   -.-...-.-.K...-.  ..J..0...0....0.
005BE0  5020D1C0 58A0D1C0 5810D208 07F1D203   D1E0C078 DE03D1E0 6001D202 6172D1E1   &.J...J...K..1K.  J.....J.-.K./.J.
005C00  F8F3D1D8 60C3D7C3 D1D8D1D8 940FD1DC   D209D1E0 C06E4110 D1E6DF09 D1E0D1DC   8.JQ-.P.JQJQ..J.  K.J.....JW..J.J.
005C20  0610925B 1000D2C7 61FAD1E2 D203B0C0   6113C940 90845810 D1CC9260 800058F0   ...$..K./.JSK...  /.. ....J..-...0
005C40  101045E0 F00C5020 D1C058A0 D1C0D201   6001C0A0 D2036003 C0825810 D20C07F1   ....0.&.J...J.K.  -...K.-.....K..1
005C60  D2016007 7C1BD27E 80006028 92408C7F   D2048050 807F5810 D1CC92F1 800058F0   K.-...K...-.. ..  K.......J..1...0
005C80  101045E0 F00C5020 D1C058B0 D1C05810   D21C07F1 0A0E50D0 50045050 D0045820   ....0.&.J...J...  K..1..&.&.&&....
005CA0  CCCC99CC 2C000779 92FF2000 9610D048   5CE0DC54 05F09120 D04B47E0 F0165B00   ................  &....0......0...
005CC0  B0495B7C BC505BE0 D084C7FE 962CD04B   4160C004 4110C004 4170C048 06700550   .....&..........  .-.............&
005CE0  5B401000 1E4B5040 1C0C8716 5C0041B0   D1BC4170 D1CF051C 58008000 1F0B5000   . ....& ....&...  J...J.........&.
005D00  800087BA 1C00D20B D2C8C05C 5B60D1C4   5B70D19C 58B0D1C0 58E0DC54 07FEF01A   ......K.K..*.-JD  ..J...J.......0.
005D20  0A37C00C 47F0F0B4 0A370000 47F0F01A   C9D1C4C6 C107C9E9 F3F99130 10024710   .....00......00.  IJDFAPIZ39......
005D40  F0240AC7 5C0EF0EC 58E01C1B 06ECD200   1C17E000 1ACC1BDC 43C01017 41E00010   0.....0.......K.  ................
005D60  43DEF0F7 19C04780 F05044E0 F0400A32   43DEF107 43C01016 19C04770 F0769180   ..07....0&..0 ..  ..1.........0...
005D80  10164780 F076920B 1028CAC0 91801002   4710F076 0A074200 10284200 10160A00   ....0...........  ..0.............
005DA0  91801002 4710F08A 0AC7D201 10281C03   92C1102B 58E0102B D2021029 101950E0   ......0...K.....  ........K.....&.
005DC0  101BC4EC 58C0F0EC 44001022 58E0F0F4   0A0007FE 91801002 4710F09E 0AC79101   ......0.......04  ..........0.....
005DE0  10194780 F0D494FE 10199102 1070C78E   47F0F0DA 91011027 07BE1200 4780F0E4   ....0M..........  .00...........0U
005E00  1BF007FF 92B8102B 0A0007FE 0000576B   C000B54B B0005C86 F2C2C1F9 F8F7F6F5   .0..............  ......*.2BA98765
005E20  F4F3C3F1 4E60F040 930BC3C8 C3BEE3AB   B39BE3B8 0319130B 47F0F014 47F0F252   43C1+-0 ..L.C...  ..T......00..02.
005E40  47F0F022 47F0F014 47F0F014 0A32C9D1   C7C6C9C5 E6E9F3F9 0009903E F26C9180   .00..00..00...IJ  GFIEWZ39....2...
005E60  1049471C F04A5BA0 102C50A0 105B4AA0   104A50A0 10609140 10274710 F1CC47F0   ....0.....&.....  ..&..-. ....1..0
005E80  F0A098BC 1C5BB6BC F0B8F0B0 1C5B9110   1C154790 F07E18D0 41A00100 1BCA47C0   0........0.&....  ....0...........
005EA0  F076D2FF DC00B000 1ADAB6EA F06441CC   00FF44C0 F25A9BAE F26C4400 105407FE   0.K.........0...  ....2...2.......
005EC0  91081C49 4780F098 94F71049 47F0F1CC   91C81044 4710F0A2 0A009150 10024710   ......0..7...01.  ......0.........
005EE0  F0AC0AC7 91011004 4780F0B8 47F0F236   91C81003 4780F0C4 91801044 4780F000   0.........0..02.  ......00.......0
005F00  58E01C44 47F0F0E4 91921003 4780F110   918C1044 4780F14C 58E01044 98ADF26C   .....00U......1.  ......1......2..
005F20  90F1F100 58101C80 C700C5FE 98F1E00C   98ADF26C 47F0F154 00000000 00000000   .11..........1..  ..2..01.........
005F40  CCCC0CCC CCCC000C 91401005 4780F15B   48B10000 4980F260 4780F13C 4B80105E   ......... ....1.  ......2-..1.....
005F60  4980F260 4720F124 4740F13C 96401C49   47F0F158 91041064 47B0F0D8 58E01050   ..2-..1.. 1.. ..  .01.......00...&
005F80  47F0F0E4 91201064 4780F15B 96101049   58B01CB0 41B000C0 50B01058 91401049   .00U.....1......  ..........&.....
005FA0  4710F178 4A80104A 50BC1060 47F0F1B4   4BC01C4A 4BC10000 1AC850C0 10609740   ..1.....&..-.01.  .....A....&..-..
005FC0  104948E0 103499E0 000843E0 F26341B0   CC051BCD 43C9103B 43DB1043 19C04770   ............2...  ................
005FE0  F1A647EF 1C3BBBE0 0C0846B0 F19C41CC   CC0147CB 103BD5C3 1C3C1036 4700F202   1...........1...  ......N.......2.
006000  96081049 9001F2B0 98ADF26C 4100F2B4   411CF264 0A029801 F2B09120 10104710   ......2...2...2.  ..2.....2.......
006020  F07E9140 10274780 F1FA94FF 102747F0   F0A09108 10494710 F21A9108 10644780   0.. ....1......0  0.......2.......
006040  F21A5BB0 1CB0D202 1C811020 50B0102C   CA0C9110 10494780 F22A9710 104947F0   2.....K......&..  ........2......0
006060  F0B856B0 1C499ABD 105B47F0 F05691B0   10264710 F1CC9120 10274710 F1CC94FE   0..........00...  ....1.......1...
006080  10499640 102747F0 F1CCD201 10581060   C7FED200 B000B000 00000001 5B5BC2D6   ... ...01.K....-  ..K.........$$BO
0060A0  D7C5D540 C0C05000 0C0C5060 C0C05768   C0C05548 B00058F6 0C000C00 00000000   PEN ..&...&.....  ........6.......
0060C0  FFC0C000 00000000 00000000 00C00000   C0C0C000 0000C9D3 FFC4E2C1 C5F0F3F1   ................  ......IL.DSAE031
0060E0  05F047F0 F01BC9D3 C2C4E2C1 C5F0F3F1   F4F3F0F3 61F0F761 F7F41B00 47F0F022   .0.00.ILBDSAE031  4303/07/74...00.
006100  41000001 05F050E0 F0FE9005 F0F24144   C0009110 4014478C F0309108 40154710   .....0&.0...02..  .... ...0... ...
006120  F03C912C 4C024780 F02C4100 001047F0   F0305811 00044154 00004850 F11A5855   0... ...0......0  0..........&1...
006140  C0004155 0C001255 4780F0AC 1B35D501   F0E43C00 4780F05A 41330002 47F0F04B   ..........0...N.  0U....0......00.
006160  5050F0EA D5033006 F0F04780 F09E4403   C0C64403 000A1852 92F05000 D2065001   &&0.N...0...0...  .........0&.K.&.
006180  5C0C1200 47B0F09A 4900F0E6 4780F092   5BC15002 47F0F09E 9A015001 47F0F09E   &.....0...0M..0.  ..&..00...&..00.
0061A0  96015000 58F0F0EE 9B05F0F2 58F0F0EA   C7FFB900 00184840 F11C1604 4110F10A   ..&...0...02.00.  ....... 1.....1.
0061C0  0A0291CE DC4B4780 FCCC5810 F11259F0   F11EC701 1CC21002 4111003C 50D1C000   .........0..1..0  1.P.........&J..
0061E0  C7FF1900 CA060000 000005F0 00018BE0   C00B46B0 F19C41CC 000142CB 1038D503   ...........0....  ....1.........N.
006200  5B5BC2C3 D6C2C5D9 00000000 00000000   5B5BC2C3 D6C2C5D9 00000000 00000000   $$BCOBER........  $$BCOBER........
```

IA-02-02-14

COBOL SYNTAX DEBUG

Practical Exercise Booklet

INSTRUCTIONS TO THE STUDENT:

This booklet contains three (3) COBOL Source program listings, each with an Associated Compiler Error Diagnostic Listing.

You are to correct the syntax errors by;

1. Circle the portion of the source statement in error,
2. pencil in the correction to be made on the source listing.

When you correct all the errors in the first practical exercise turn to the back of this booklet and check your answers against the school solution. If you have any questions concerning this exercise see your instructor for assistance.

Follow this procedure for all three practical exercises in this booklet.

IA-01-03-03

308

```
CB545 V2 LVL78 01MAY72                    IBM OS AMERICAN NATIONAL STANDARD C

   1

00001          IDENTIFICATION DIVISION.
00002          PROGRAM-ID.    COBSYN1.
00003          AUTHOR.        SP6 DW GROSS, SOFTWARE.
00004          INSTALLATION.  CSD, USAIA, FT HARRISON, IN  46216.
00005          DATE-WRITTEN.  MAR 3 1978.
00006          DATE-COMPILED. JUN 15,1978
00007          SECURITY.      UNCLASSIFIED.
00008         *REMARKS.       THIS PROGRAM LISTS ALL ORDERS RECEIVED
00009         *               EACH DAY. IT LISTS THE PART NUMBER, CUSTOMER
00010         *               NAME, AND CUSTOMER ADDRESS.
00011         *
00012         *               WS-ORDER-EOF-SW IS USED IN THE PROCEDURE
00013         *               DIVISION TO SIGNIFY WHEN THE END OF FILE
00014         *               CONDITION HAS BEEN REACHED.
00015          ENVIRONMENT DIVISION.
00016          CONFIGURATION SECTION.
00017          SOURCE-COMPUTER. IBM-360-H40.
00018          OBJECT-COMPUTER. IBM-360-H40.
00019          SPECIAL-NAMES.
00020              C01 IS TOP-OF-PAGE.
00021          INPUT-OUTPUT SECTION.
00022          FILE-CONTROL.
00023              SELECT IM-ORDER-FILE
00024                  ASSIGN TO UT-S-SYS006.
00025              SELECT OR-ORDERS-LISTING
00026                  ASSIGN TO UT-S-SYS005.
00027          DATA DIVISION.
00028          FILE SECTION.
00029          FD  IM-ORDER-FILE,
00030              BLOCK CONTAINS 1 RECORDS,
00031              RECORD CONTAINS 80 CHARACTERS,
00032              LABEL RECORD IS STANDARD.
00033              DATA RECORD IS IM01-ORDER-RCD.
00034          01  IM01-ORDER-RCD.
00035              05  IM01-PART-NO                   PIC 9(4).
00036              05  IM01-CUSTOMER-NAME             PIC X(18).
00037              05  IM01-CUSTOMER-ADDRESS          PIC X(18).
00038              05  IM01-UNIT-OF-ISSUE             PIC XX.
00039              05  IM01-QTY-ORDERED               PIC 999.
00040              05  IM01-PRICE-ITEM                PIC 9(3)V99.
00041              05  IM01-CUSTOMER-CODE             PIC X.
00042              05  FILLER                         PIC X(31).
00043          FD  OR-ORDERS-LISTING,
00044              BLOCK CONTAINS 1 RECORDS,
00045              RECORD CONTAINS 133 CHARACTERS,
00046              LABEL RECORD IS STANDARD,
00047              DATA RECORD IS OR01-ORDER-LINE.
00048          01  OR01-ORDER-LINE                    PIC X(133).
00049          WORKING-STORAGE SECTION.
00050          77  WS-ORDER-EOF-SW                    PIC XXX
00051                          VALUE 'OFF'.
00052          01  WS-HEADING-LINE.
00053              05  FILLER                         PIC X(33)
00054                          VALUE SPACES.
```

```
    05  FILLER                          PIC X(7)
                    VALUE 'PART NO'.
    05  FILLER                          PIC X(11)
                    VALUE SPACES.
    05  FILLER                          PIC X(13)
                    VALUE 'CUSTOMER NAME'.
    05  FILLER                          PIC X(12)
                    VALUE SPACES.
    05  FILLER                          PIC X(16)
                    VALUE 'CUSTOMER ADDRESS'.
    05  FILLER                          PIC X(41)
                    VALUE SPACES.
01  WS-DETAIL-LINE.
    05  FILLER                          PIC X(35)
                    VALUE SPACES.
    05  WS-DL-PART-NO                   PIC 9(4).
    05  FILLER                          PIC 1(X0)
                    VALUE SPACES.
    05  WS-DL-CUSTOMER-NAME             PIC X(18).
    05  FILLER                          PIC X(8)
                    VALUE SPACES.
    05  WS-DL-CUSTOMER-ADDRESS          PIC X(18).
    05  FILLER                          PIC X(40)
                    VALUE SPACES
PROCEDURE DIVISION.
0010-DRIVER.
    OPEN INPUT IM-ORDER-FILE
        OUTPUT OR-ORDERS-LISTING.
    WRITE OR01-ORDER-LINE FROM WS-HEADING-LINE
        AFTER ADVANCING TOP-OF-PAGE.
    PERFORM 0020-READ-LIST-RTN THRU 0020-EIXT
        UNTIL WS-ORDER-EOF-SW EQUAL 'ON'.
    CLOSE IM-ORDER-FILE
          OR-ORDERS-LISTING.
0010-EXIT.
    STOP RUN.
0020-READ-LIST-RTN.
    READ IM-ORDER-FILE
        AT END
            MOVE 'ON' TO WS-ORDER-EOF-SW
            GO TO 0020-EXIT.
    MOVE IM01-PART-NO TO WS-DL-PART-NU.
    MOVE IM01-CUSTOMER-NAME TO WS-DL-CUSTOMER-NAME.
    MOVE IM01-CUSTOMER-ADDRESS TO WS-DL-CUSTOMER-ADDRESS.
    WRITE OR01-ORDER-LINE FROM WS-DETAIL-LINE
        AFTER ADVANCING 2 LINE.
0020-EXIT.
    EXIT.
```

```
4

CARD   ERROR MESSAGE

33     IKF1004I-E     INVALID WORD DATA . SKIPPING TO NEXT RECOGNIZABLE WORD.
29     IKF2146I-W     RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING
                      MAXIMUM COMPUTED SIZE.
71     IKF2039I-C     PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST',
                      THEN L(6)BDZ9BDZ9.
71     IKF2129I-C     VALUE CLAUSE LITERAL DOES NOT CONFORM TO PICTURE. CHANGED TO ZERO.
79     IKF1043I-W     END OF SENTENCE SHOULD PRECEDE PROCEDURE . ASSUMED PRESENT.
85     IKF3001I-E     0020-EIXT NOT DEFINED.  THRU OPTION DISCARDED.
96     IKF3001I-E     WS-DL-PART-NU NOT DEFINED. DISCARDED.
99     IKF4003I-E     EXPECTING NEW STATEMENT. FOUND LINES . DELETING TILL NEXT VERB OR PROCEDURE-NAME.
```

3

IA-01-03-03

```
 PROGRAM-ID.    COBSYN2.
 IDENTIFICATION DIVISION.
 AUTHOR.        SP6 DW GROSS.
 INSTALLATION.  CSD, USAIA, FT HARRISON, IN  46216.
 DATE WRITTEN.  MAR 3, 1978.
 DATE-COMPILED. JUN 15,1978
 SECURITY.      UNCLASSIFIED.
*REMARKS.       THIS PROGRAM LISTS ALL PERSONNEL IN THE
*               BATTALION WITH OVER 75 MONTHS SERVICE.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-360-H40.
 OBJECT-COMPUTER. IBM-360-H40.
 SPECIAL-NAMES.
    C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT IM-PERSONNEL-MASTER-FILE
       ASSIGN TO TU-S-SYS006.
    SELECT OR-PRINTER
       ASSIGN TO UT-S-SYS005.
 DATA DIVISION.
 FILE SECTION.
 FD  IM-PERSONNEL-MASTER-FILE
    BLOCK CONTAINS 1 RECORDS,
    RECORD CONTAINS 80 CHARACTERS,
    LABEL RECORD IS STANDARD,
    DATA RECORD IS IM01-PERSONNEL-MASTER-RCD.
 01  IM01-PERSONNEL-MASTER-RCD.
    05  IM01-BATTLAION                         PIC XX.
    05  IM01-COMPANY                           PIC X.
    05  IM01-SSAN                              PIC 9(9).
    05  IM01-NAME                              PIC X(18).
    05  IM01-PAY-GRADE                         PIC XY.
    05  IM01-RANK                              PIC X(3).
    05  IM01-ETS                               PIC X(6).
    05  IM01-DATE-OF-ENTRY                     PIC X(6).
    05  IM01-MONTHS-OF-SERVICE                 PIC 9(3).
    05  IM01-NUMBER-OF-DEPENDENTS              PIC 99.
    05  IM01-PREVIOUS-ASSIGN-INFO              PIC X(28).
 FD  OR-PRINTER,
    BLOCK CONTAINS 1 RECORDS,
    RECORD CONTAINS 133 CHARACTERS,
    LABEL RECORD IS STANDARD,
    DATA RECORD IS OR01-PRINTER-RCD.
 01  OR01-PRINTER-RCD                          PIC X(13X).
 WORKING-STORAGE SECTION.
 77  WS-MASTER-FILE-EOF-SWITCH                 PIC XXX
                VALUE 'OFF'
 01  WS-HEADING-LINE.
    05  FILLER                                 PIC X(53)
                VALUE SPACES.
    05  FILLER                                 PIC X(5)
```

4

IA-01-03-03

```
                    VALUE 'OVER 75 MONTHS OF SERVICE'.
    05  FILLER                          PIC X(55)
                    VALUE SPACES
01  WS-DETAIL-LINE.
    05  FILLER                          PIC X(36)
                    VALUE SPACES.
    05  WS-DL-NAME                      PIC X(18).
    05  FILLER                          PIC X(10)
                    VALUE SPACES.
    05  WS-DL-RANK                      PIC XXX.
    05  FILLER                          PIC X(9)
                    VALUE SPACES.
    05  WS-DL-SSAN                      PIC 9(9).
    05  FILLER                          PIC X(7)
                    VALUE SPACES.
    05  WS-DL-MONTHS-OF-SERVICE         PIC ZZ9.
    05  FILLER                          PIC X(38).
                    VALUE SPACES
PROCEDURE DIVISION.
0010-DRIVER.
    OPEN INPUT IM-PERSONNEL-MASTER-FILE
         OUTPUT OR-PRINTER.
    WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE
        AFTER ADVANCING TOP-OF-PAGE.
    PERFORM 002/-READ-LIST-RTN THRU 0020-EXIT
        UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
    CLOSE IM-PERSONNEL-MASTER-FILE.
          OR-PRINTER.
0010-EXIT.
    STOP RUN.
0020-READ-LIST-RTN.
    READ IM-PERSONNEL-MASTER-FILE
        AT END
            MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
            GO TO 0020-EXIT.
    IF IM01-MONTHS-OF-SERVICE GREATER THAN 75
        THEN
            MOVE IM01-NAME TO WS-DL-NAME
            MOVE IM01-RANK TO WS-DL-RANK
            MOVE IM01-SSAN TO WS-DL-SSAN
            MOVE WS-DL-MONTHS-OF-SERVICE TO
                IM01-MONTHS-OF-SERVICE
            WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
                AFTER ADVANCING 2 LINES.
0020-EXIT.
    EXIT.
```

```
CARD    ERROR MESSAGE

1       IKF1129I-C     ID DIV. HEADER EXTRANEOUS, MISSING OR MISPLACED. ONE ASSUMED PRESENT.
5       IKF1087I-W     ' DATE ' SHOULD NOT BEGIN A-MARGIN.
20      IKF1155I-W     DEVICE CLASS INVALID IN SYSTEM-NAME. SKIPPING TO NEXT FIELD.
35      IKF2039I-C     PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST',
                       THEN L(6)BDZ9BDZ9.
19      IKF2146I-W     RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING
                       MAXIMUM COMPUTED SIZE.
47      IKF2039I-C     PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST',
                       THEN L(6)BDZ9BDZ9.
51      IKF1043I-W     END OF SENTENCE SHOULD PRECEDE 01 . ASSUMED PRESENT.
21      IKF2146I-W     RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING
                       MAXIMUM COMPUTED SIZE.
54      IKF2126I-C     VALUE CLAUSE LITERAL TOO LONG. TRUNCATED TO PICTURE SIZE.
58      IKF1043I-W     END OF SENTENCE SHOULD PRECEDE 01 . ASSUMED PRESENT.
72      IKF1004I-E     INVALID WORD VALUE . SKIPPING TO NEXT RECOGNIZABLE WORD.
79      IKF1007I-W     SLASH NOT PRECEDED BY A SPACE. ASSUME SPACE.
79      IKF3001I-E     002 NOT DEFINED. STATEMENT DISCARDED.
79      IKF1007I-W     MINUS SIGN NOT PRECEDED BY A SPACE. ASSUME SPACE.
79      IKF3001I-E     READ-LIST-RTN NOT DEFINED.
82      IKF1086I-W     OR-PRINTER SHOULD BEGIN A-MARGIN.
95      IKF4052I-E     DNM=1-358  (ED) MAY NOT BE TARGET FIELD FOR DNM=2-283  (NE) IN MOVE STATEMENT, AND
                       IS DISCARDED.
```

314

```
 IDENTIFICATION DIVISION.
 PROGRAM ID.    COBSYN3.
 AUTHOR.        SP6 DW GROSS.
 INSTALLATION.  CSD, USAIA, FT HARRISON, IN  46216.
 DATE-WRITTEN.  MAR 3, 1978.
 DATE-COMPILED. JUN 15,1978
 SECURITY.      UNCLASSIFIED.
*REMARKS.       THIS PROGRAM PRODUCES THE DEPENDENT INFORMATION
*               REPORT. IT LISTS ALL PERSONNEL IN THE BATTALION
*               WITH 5 OR MORE DEPENDENTS BY NAME, RANK, COMPANY,
*               AND SIZE OF FAMILY.
*
*               WS-MASTER-FILE-EOF-SWITCH IS USED TO SIGNIFY
*               WHEN THE END OF FILE CONDITION HAS BEEN
*               ENCOUNTERED.
*               WS-TOTAL-DEPENDENTS IS USED TO ACCUMULATE THE
*               DEPENDENTS FOR THE PURPOSE OF PRINTING IT IN
*               THE TOTAL LINE.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE COMPUTER. IBM-360-H40.
 OBJECT-COMPUTER. IBM-360-H40.
 SPECIAL-NAMES.
     C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE CONTROL.
     SELECT IM-PERSONNEL-MASTER-FILE
         ASSIGN TO UT-S-SYS006.
     SELECT OR-PRINTER
         ASSIGN TO UT-S-SYS005.
 DATA DIVISION.
 FILE SECTION.
 FD  IM-PERSONNEL-MASTER-FILE,
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 80 CHARACTERS,
     LABEL RECORD IS STANDARD,
     DATA RECORD IS IM01-PERSONNEL-MASTER-RCD.
 01  IM01-PERSONNEL-MASTER-RCD
     05  IM01-BATTALION                  PIC XX.
     05  IM01-COMPANY                    PIC X.
     05  IM01-SSAN                       PIC 9(9).
     05  IM01-NAME                       PIC X(18).
     05  IM01-PAY-GRADE                  PIC XX.
     05  IM01-RANK                       PIC XXX.
     05  IM01-ETS                        PIC X(6).
     05  IM01-DATE-OF-ENTRY              PIC X(6).
     05  IM01-MONTHS-OF-SERVICE          PIC 999.
     05  IM01-NUMBER-OF-DEPENDENTS       PIC XX.
     05  IM01-PREVIOUS-ASSIGN-INFO       PIC X(28).
 FD  OR-PRINTER,
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 133 CHARACTERS,
     LABEL RECORD IS STANDARD,
```

7 IA-01-C3-C3

```
00055             DATA RECORD IS OR01-PRINTER-RCD.
00056         01  OR01-PRINTER-RCD                    PIC X(133).
00057         WORKING-STORAGE SECTION.
00058         77  WS-LINE-COUNTER                     PIC Z9
00059                         VALUE ZEROS.
00060         77  WS-TOTAL-DEPENDENTS                 PIC ZZ9
00061                         VALUE ZEROS.
00062         77  WS-MASTER-FILE-EOF-SWITCH           PIC XXX
00063                         VALUE 'OFF'.
00064         01  WS-HEADING-LINE-1.
00065             05  FILLER                          PIC X(53)
00066                         VALUE SPACES.
00067             05  FILLER                          PIC X(28)
00068                         VALUE 'DEPENDENT INFORMATION REPORT'.
00069             05  FILLER                          PIC X(52)
00070                         VALUE SPACES.
00071         01  WS-HEADING-LINE-2.
00072             05  FILLER                          PIC X(38)
00073                         VALUE SPACES.
00074             05  FILLER                          PIC X(4).
00075                         VALUE 'NAME'.
00076             05  FILLER                          PIC X(18)
00077                         VALUE SPACES.
00078             05  FILLER                          PIC X(4)
00079                         VALUE 'RANK'.
00080         05  FILLER                              PIC X(10)
00081                         VALUE SPACES.
00082             05  FILLER                          PIC X(7)
00083                         VALUE 'COMPANY'.
00084             05  FILLER                          PIC X(7)
00085                         VALUE SPACES.
00086             05  FILLER                          PIC X(11)
00087                         VALUE 'FAMILY SIZE'.
00088             05  FILLER                          PIC X(34)
00089                         VALUE SPACES.
00090         01  WS-DETAIL-LINE.
00091             05  FILLER                          PIC X(32)
00092                         VALUE SPACES.
00093             05  WS-DL-NAMEC                     PIC X(18)
00094             05  FILLER                          PIC X(10)
00095                         VALUE SPACES.
00096             05  WS-DL-RANK                      PIC XXX.
00097             05  FILLER                          PIC X(14)
00098                         VALUE SPACES.
00099             05  WS-DL-COMPANY                   PIC X.
00100             05  FILLER                          PIC X(15)
00101                         VALUE SPACE..
00102             05  WS-DL-FAMILY-SIZE               PIC ZZZ9.
00103             05  FILLER                          PIC X(36)
00104                         VALUE SPACES.
00105         01  WS-TOTAL-LINE.
00106             05  FILLER                          PIC X(32)
00107                         VALUE SPACES.
00108             05  FILLER                          PIC X(34)
00109                         VALUE 'TOTAL NUMBER OF DEPENDENTS WITHIN'.
00110             05  FILLER                          PIC X(18)
00111                         VALUE 'THE BATTALION IS'.
```

IA-01-03-03

```
    05  WS-TL-NUMBER-OF-DEPENDETS       PIC ZZ9.
    05  FILLER                          PIC X(46)
                    VALUE SPACES.
PROCEDURE DIVISION.
0010-DRIVER.
    OPEN INPUT IM-PERSONNEL-MASTER-FILE
        OUTPUT OR-PRINTER.
    PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
    PERFORM 0020-READ-WRITE-RTN THRU 0020-EXIT
        UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
    MOVE WS-TOTAL-DEPENDETNS TO WS-TL-NUMBER-OF-DEPENDENTS.
    WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
        AFTER ADVANCING 3 LINES.
    CLOSE IM-PERSONNEL-MASTER-FILE,
          OR-PRINTER.
0010-EXIT.
    EXIT.
0020-READ-WRITE-RTN.
    READ IM-PERSONNEL-MASTER-FILE
        AT END
            MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
            GO TO 0020-EXIT.
    ADD IM01-NUMBER-OF-DEPENDENTS TO WS-TOTAL-DEPENDENTS.
    IF IM01-NUMBER-OF-DEPENDENTS LESS THAN 4
        THEN
            GO TO 0020-EXIT.
    IF WS-LINE-COUNTER GREATER THAN 15
        THEN
            PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
    MOVE IM01-NAME TO WS-DL-NAME.
    MOVE IM01-RANK TO WS-DL-RANK.
    MOVE IM01-COMPANY TO WS-DL-COMPANY.
    ADD 1, IM01-NUMBER-OF-DEPENDENTS GIVING WS-DL-FAMILY-SIZE.
    WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
        AFTER ADVANCING 2 LINES.
    ADD 1 TO WS-LINE-COUNTER.
0020-EXIT.
    EXIT.
0030-HEADING-RTN.
    MOVE 0 TO WS-LINE-COUNTER.
    WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-1
        AFTER ADVANCING TOP-OF-PAGE.
    WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-2
        AFTER ADVANCING 3 LINES.
0030-EXIT.
    EXIT.
```

9

IA-01-03-03

```
  9

CARD   ERROR MESSAGE

2      IKF1087I-W     ' PROGRAM ' SHOULD NOT BEGIN A-MARGIN.
3      IKF1097I-E     PROGRAM-ID MISSING OR MISPLACED. IF PROGRAM-ID DOES NOT IMMEDIATELY FOLLOW
                      IDENTIFICATION DIVISION, IT WILL BE IGNORED.
22     IKF1087I-W     ' SOURCE ' SHOULD NOT BEGIN A-MARGIN.
22     IKF1004I-E     INVALID WORD SOURCE . SKIPPING TO NEXT RECOGNIZABLE WORD.
27     IKF1087I-W     ' FILE ' SHOULD NOT BEGIN A-MARGIN.
27     IKF1004I-E     INVALID WORD FILE . SKIPPING TO NEXT RECOGNIZABLE WORD.
28     IKF1003I-W     FILE-CONTROL PARAGRAPH NAME MISSING. ASSUMED PRESENT.
40     IKF1043I-W     END OF SENTENCE SHOULD PRECEDE 05 . ASSUMED PRESENT.
75     IKF1004I-E     INVALID WORD VALUE . SKIPPING TO NEXT RECOGNIZABLE WORD.
94     IKF1043I-W     END OF SENTENCE SHOULD PRECEDE 05 . ASSUMED PRESENT.
101    IKF1081I-W     PERIOD NOT FOLLOWED BY SPACE. ASSUME END OF SENTENCE.
101    IKF1080I-W     PERIOD PRECEDED BY SPACE. ASSUME END OF SENTENCE.
122    IKF3001I-E     WS-TOTAL-DEPENDETNS NOT DEFINED. DISCARDED.
122    IKF3001I-E     WS-TL-NUMBER-OF-DEPENDENTS NOT DEFINED.
134    IKF4019I-E     DNM=1-427  (AN) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT.
                      ARBITRARILY SUBSTITUTING TALLY .
134    IKF4019I-E     DNM=2-81  (NE) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT. ARBITRARILY
                      SUBSTITUTING TALLY .
141    IKF3001I-E     WS-DL-NAME NOT DEFINED. DISCARDED.
144    IKF4019I-E     DNM=1-427  (AN) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT.
                      ARBITRARILY SUBSTITUTING TALLY .
144    IKF5011I-W     AN INTERMEDIATE RESULT OR A SENDING FIELD MIGHT HAVE ITS HIGH ORDER DIGIT POSITION
                      TRUNCATED.
147    IKF4019I-E     DNM=2-49  (NE) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT. ARBITRARILY
                      SUBSTITUTING TALLY .
```

333

10

IA-01-03-03

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    COBSYN1.
       AUTHOR.        SP6 DW GROSS, SOFTWARE.
       INSTALLATION.  CSD, USAIA, FT HARRISON, IN  46216.
       DATE-WRITTEN.  MAR 3 1978.
       DATE-COMPILED. JUN 21,1978
       SECURITY.      UNCLASSIFIED.
      *REMARKS.       THIS PROGRAM LISTS ALL ORDERS RECEIVED
      *               EACH DAY. IT LISTS THE PART NUMBER, CUSTOMER
      *               NAME, AND CUSTOMER ADDRESS.
      *
      *               WS-ORDER-EOF-SW IS USED IN THE PROCEDURE
      *               DIVISION TO SIGNIFY WHEN THE END OF FILE
      *               CONDITION HAS BEEN REACHED.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-360-H40.
       OBJECT-COMPUTER. IBM-360-H40.
       SPECIAL-NAMES.
           C01 IS TOP-OF-PAGE.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT IM-ORDER-FILE
               ASSIGN TO UT-S-SYS006.
           SELECT OR-ORDERS-LISTING
               ASSIGN TO UT-S-SYS005.
       DATA DIVISION.
       FILE SECTION.
       FD  IM-ORDER-FILE,
           BLOCK CONTAINS 1 RECORDS,
           RECORD CONTAINS 80 CHARACTERS,
           LABEL RECORD IS STANDARDⓈ,
           DATA RECORD IS IM01-ORDER-RCD.
       01  IM01-ORDER-RCD.
           05  IM01-PART-NO                    PIC 9(4).
           05  IM01-CUSTOMER-NAME              PIC X(18).
           05  IM01-CUSTOMER-ADDRESS           PIC X(18).
           05  IM01-UNIT-OF-ISSUE              PIC XX.
           05  IM01-QTY-ORDERED                PIC 999.
           05  IM01-PRICE-ITEM                 PIC 9(3)V99.
           05  IM01-CUSTOMER-CODE              PIC X.
           05  FILLER                          PIC X(㉛). (29)
       FD  OR-ORDERS-LISTING,
           BLOCK CONTAINS 1 RECORDS,
           RECORD CONTAINS 133 CHARACTERS,
           LABEL RECORD IS STANDARD,
           DATA RECORD IS OR01-ORDER-LINE.
       01  OR01-ORDER-LINE                     PIC X(133).
       WORKING-STORAGE SECTION.
       77  WS-ORDER-EOF-SW                     PIC XXX
                           VALUE 'OFF'.
       01  WS-HEADING-LINE.
           05  FILLER                          PIC X(33)
                           VALUE SPACES.
```

```
    05  FILLER                              PIC X(7)
                    VALUE 'PART NO'.
    05  FILLER                              PIC X(11)
                    VALUE SPACES.
    05  FILLER                              PIC X(13)
                    VALUE 'CUSTOMER NAME'.
    05  FILLER                              PIC X(12)
                    VALUE SPACES.
    05  FILLER                              PIC X(16)
                    VALUE 'CUSTOMER ADDRESS'.
    05  FILLER                              PIC X(41)
                    VALUE SPACES.
01  WS-DETAIL-LINE.
    05  FILLER                              PIC X(35)
                    VALUE SPACES.
    05  WS-DL-PART-NO                       PIC 9(4).
    05  FILLER                              PIC 1(X0) X(10)
                    VALUE SPACES.
    05  WS-DL-CUSTOMER-NAME                 PIC X(18).
    05  FILLER                              PIC X(8)
                    VALUE SPACES.
    05  WS-DL-CUSTOMER-ADDRESS              PIC X(18).
    05  FILLER                              PIC X(40)
                    VALUE SPACESO.
PROCEDURE DIVISION.
0010-DRIVER.
    OPEN INPUT IN-ORDER-FILE
        OUTPUT OR-ORDERS-LISTING.
    WRITE OR01-ORDER-LINE FROM WS-HEADING-LINE
        AFTER ADVANCING TOP-OF-PAGE.
    PERFORM 0020-READ-LIST-RTN THRU 0020-EIXT EXIT
        UNTIL WS-ORDER-EOF-SW EQUAL 'ON'.
    CLUSE IN-ORDER-FILE
          OR-ORDERS-LISTING.
0010-EXIT.
    STOP RUN.
0020-READ-LIST-RTN.
    READ IN-ORDER-FILE
        AT END
            MOVE 'ON' TO WS-ORDER-EOF-SW
            GO TO 0020-EXIT.
    MOVE IM01-PART-NO TO WS-DL-PART-NO NO.
    MOVE IM01-CUSTOMER-NAME TO WS-DL-CUSTOMER-NAME.
    MOVE IM01-CUSTOMER-ADDRESS TO WS-DL-CUSTOMER-ACDRESS.
    WRITE OR01-ORDER-LINE FROM WS-DETAIL-LINE
        AFTER ADVANCING 2 LINEO LINES.
0020-EXIT.
    EXIT.
```

```
 PROGRAM-ID.      C[illegible]YN2.   IDENTIFICATION DIVISION.
 IDENTIFICATION [illegible]ISION.   PROGRAM-ID.    COBSHN2.
 AUTHOR.          SP6 DW GROSS.
 INSTALLATION.    CSD, USAIA, FT HARRISON, IN  46216.
 DATE WRITTEN.    MAR 3, 1978. DATE-WRITTEN.
 DATE-COMPILED.   JUN 21,1978
 SECURITY.        UNCLASSIFIED.
*REMARKS.         THIS PROGRAM LISTS ALL PERSONNEL IN THE
*                 BATTALION WITH OVER 75 MONTHS SERVICE.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-360-H40.
 OBJECT-COMPUTER. IBM-360-H40.
 SPECIAL-NAMES.
     C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT IN-PERSONNEL-MASTER-FILE
         ASSIGN TO U-S-SYS006.      UT-S-SYS006.
     SELECT OR-PRINTER
         ASSIGN TO UT-S-SYS005.
 DATA DIVISION.
 FILE SECTION.
 FD  IN-PERSONNEL-MASTER-FILE
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 80 CHARACTERS,
     LABEL RECORD IS STANDARD,
     DATA RECORD IS IM01-PERSONNEL-MASTER-RCD.
 01  IM01-PERSONNEL-MASTER-RCD.
     05  IM01-BATTLAION                          PIC XX.
     05  IM01-COMPANY                            PIC X.
     05  IM01-SSAN                               PIC 9(9).
     05  IM01-NAME                               PIC X(18).
     05  IM01-PAY-GRADE                          PIC XY. XX.
     05  IM01-RANK                               PIC X(3).
     05  IM01-ETS                                PIC X(6).
     05  IM01-DATE-OF-ENTRY                      PIC X(6).
     05  IM01-MONTHS-OF-SERVICE                  PIC 9(3).
     05  IM01-NUMBER-OF-DEPENDENTS               PIC 99.
     05  IM01-PREVIOUS-ASSIGN-INFO               PIC X(20).
 FD  OR-PRINTER,
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 133 CHARACTERS,
     LABEL RECORD IS STANDARD,
     DATA RECORD IS OR01-PRINTER-RCD.
 01  OR01-PRINTER-RCD                            PIC X(13X). X(133).
 WORKING-STORAGE SECTION.
 77  WS-MASTER-FILE-EOF-SWITCH                   PIC XXX
                         VALUE 'OFF'                  VALUE 'OFF'.
 01  WS-HEADING-LINE.
     05  FILLER                                  PIC X(53)
                         VALUE SPACES.
     05  FILLER                                  PIC X(5)  X(25)
```

13

IA-01-03-03

6

```
                               VALUE 'OVER 75 MONTHS OF SERVICE'.
           05  FILLER                                PIC X(55)
                               VALUE SPACES
       01  WS-DETAIL-LINE.
           05  FILLER                                PIC X(36)
                               VALUE SPACES.
           05  WS-DL-NAME                            PIC X(18).
           05  FILLER                                PIC X(10)
                               VALUE SPACES.
           05  WS-DL-RANK                            PIC XXX.
           05  FILLER                                PIC X(9)
                               VALUE SPACES.
           05  WS-DL-SSAN                            PIC 9(9).
           05  FILLER                                PIC X(7)
                               VALUE SPACES.
           05  WS-DL-MONTHS-OF-SERVICE               PIC ZZ9.
           05  FILLER                                PIC X(38).
                               VALUE SPACES
       PROCEDURE DIVISION.
       0010-DRIVER.
           OPEN INPUT IM-PERSONNEL-MASTER-FILE
                OUTPUT OR-PRINTER.
           WRITE CR01-PRINTER-RCD FROM WS-HEADING-LINE
               AFTER ADVANCING TOP-OF-PAGE.
           PERFORM 002/-READ-LIST-RTN THRU 0020-EXIT 0020-READ-LIST-RTN
               UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
           CLOSE IM-PERSONNEL-MASTER-FILE   CLOSE IM-PERSONNEL-MASTER-FILE
                 OR-PRINTER                        OR-PRINTER.
       0010-EXIT.
           STOP RUN.
       0020-READ-LIST-RTN.
           READ IM-PERSONNEL-MASTER-FILE
               AT END
                   MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
                   GO TO 0020-EXIT.
           IF IM01-MONTHS-OF-SERVICE GREATER THAN 75
               THEN
                   MOVE IM01-NAME TO WS-DL-NAME
                   MOVE IM01-RANK TO WS-DL-RANK
                   MOVE IM01-SSAN TO WS-DL-SSAN
                   MOVE WS-DL-MONTHS-OF-SERVICE TO   move IM01-MONTHS-OF-SERVICE TO
                       IM01-MONTHS-OF-SERVICE          WS-DL-MONTHS-OF-SERVICE
                   WRITE CR01-PRINTER-RCD FROM WS-DETAIL-LINE
                       AFTER ADVANCING 2 LINES.
       0020-EXIT.
           EXIT.
```

```
IDENTIFICATION DIVISION.   PROGRAM-ID.
PROGRAM-ID,    COBSYN3.
AUTHOR.        SP6 DW GROSS.
INSTALLATION.  CSD, USAIA, FT HARRISON, IN  46216.
DATE-WRITTEN.  MAR 3, 1978.
DATF-COMPILED. JUN 21,1978
SECURITY.      UNCLASSIFIED.
*REMARKS.      THIS PROGRAM PRODUCES THE DEPENDENT INFORMATION
*              REPORT. IT LISTS ALL PERSONNEL IN THE BATTALION
*              WITH 5 OR MORE DEPENDENTS BY NAME, RANK, COMPANY,
*              AND SIZE OF FAMILY.
*
*              WS-MASTER-FILE-EOF-SWITCH IS USED TO SIGNIFY
*              WHEN THE END OF FILE CONDITION HAS BEEN
*              ENCOUNTERED.
*              WS-TOTAL-DEPENDENTS IS USED TO ACCUMULATE THE
*              DEPENDENTS FOR THE PURPOSE OF PRINTING IT IN
*              THE TOTAL LINE.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE COMPUTER. IBM-360-H40.  SOURCE-COMPUTER.
 OBJECT-COMPUTER. IBM-360-H40.
 SPECIAL-NAMES.
     C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE CONTROL.
     SELECT IM-PERSONNEL-MASTER-FILE
         ASSIGN TO UT-S-SYS006.
     SELECT OR-PRINTER
         ASSIGN TO UT-S-SYS005.
 DATA DIVISION.
 FILE SECTION.     FILE-SECTION.
 FD  IM-PERSONNEL-MASTER-FILE,
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 80 CHARACTERS,
     LABEL RECORD IS STANDARD,
     DATA RECORD IS IM01-PERSONNEL-MASTER-RCD.
 01  IM01-PERSONNEL-MASTER-RCD            IM01-PERSONNEL-MASTER-RCD.
     05  IM01-BATTALION                  PIC XX.
     05  IM01-COMPANY                    PIC X.
     05  IM01-SSAN                       PIC 9(9).
     05  IM01-NAME                       PIC X(18).
     05  IM01-PAY-GRADE                  PIC XX.
     05  IM01-RANK                       PIC XXX.
     05  IM01-ETS                        PIC X(6).
     05  IM01-DATE-OF-ENTRY              PIC X(6).
     05  IMC1-MONTHS-OF-SERVICE          PIC 999.
     05  IM01-NUMBER-OF-DEPENDENTS       PIC XX.   99.
     05  IM01-PREVIOUS-ASSIGN-INFO       PIC X(28).
 FD  OR-PRINTER,
     BLOCK CONTAINS 1 RECORDS,
     RECORD CONTAINS 133 CHARACTERS,
     LABEL RECORD IS STANDARD,
           DATA RECORD IS OR01-PRINTER-RCD.
       01  OR01-PRINTER-RCD                      PIC X(133).
       WORKING-STORAGE SECTION.
       77  WS-LINE-COUNTER                       PIC Z9     99
                           VALUE ZEROS.
       77  WS-TOTAL-DEPENDENTS                   PIC ZZ9   999
                           VALUE ZEROS.
       77  WS-MASTER-FILE-EOF-SWITCH             PIC XXX
                           VALUE 'OFF'.
       01  WS-HEADING-LINE-1.
           05  FILLER                            PIC X(53)
                           VALUE SPACES.
           05  FILLER                            PIC X(28)
                           VALUE 'DEPENDENT INFORMATION REPORT'.
           05  FILLER                            PIC X(52)
                           VALUE SPACES.
       01  WS-HEADING-LINE-2.
           05  FILLER                            PIC X(38)
                           VALUE SPACES.
           05  FILLER                            PIC X(4.          PIC X(4)
                           VALUE 'NAME'.
           05  FILLER                            PIC X(18)
                           VALUE SPACES.
           05  FILLER                            PIC X(4)
                           VALUE 'RANK'.
       05  FILLER                                PIC X(10)
                           VALUE SPACES.
           05  FILLER                            PIC X(7)
                           VALUE 'COMPANY'.
           05  FILLER                            PIC X(7)
                           VALUE SPACES.
           05  FILLER                            PIC X(11)
                           VALUE 'FAMILY SIZE'.
           05  FILLER                            PIC X(34)
                           VALUE SPACFS.
       01  WS-DETAIL-LINE.
           05  FILLER                            PIC X(32)
                           VALUE SPACES.
           05  WS-DL-NAMEC                       PIC X(18(  PIC X(18).    WS-DL-NAME
           05  FILLER                            PIC X(10)
                           VALUE SPACES.
           05  WS-DL-RANK                        PIC XXX.
           05  FILLER                            PIC X(14)
                           VALUE SPACES.
           05  WS-DL-COMPANY                     PIC X.
           05  FILLER                            PIC X(15)
                           VALUE SPACE..                     SPACES.
           05  WS-DL-FAMILY-SIZE                 PIC ZZZ9.
           05  FILLER                            PIC X(36)
                           VALUE SPACES.
       01  WS-TOTAL-LINE.
           05  FILLER                            PIC X(32)
                           VALUE SPACES.
           05  FILLER                            PIC X(34)
                           VALUE 'TOTAL NUMBER OF DEPENDENTS WITHIN'.
           05  FILLER                            PIC X(18)
                           VALUE 'THE BATTALION IS'.
```

16

IA-01-03-03

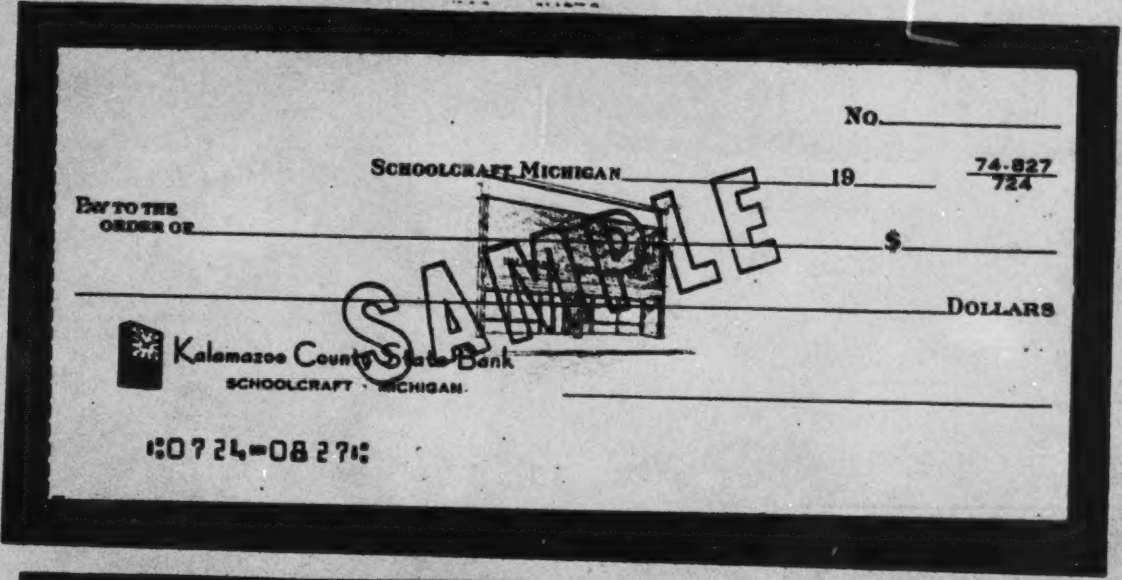
3

```
               05  WS-TL-NUMBER-OF-DEPENDETS           PIC ZZ9.
               05  FILLER                              PIC X(46)
                             VALUE SPACES.
       PROCEDURE DIVISION.
       0010-DRIVER.
           OPEN INPUT IM-PERSONNEL-MASTER-FILE
               OUTPUT OR-PRINTER.
           PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
           PERFORM 0020-READ-WRITE-RTN THRU 0020-EXIT
               UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
           MOVE WS-TOTAL-DEPENDETNS TO WS-TL-NUMBER-OF-DEPENDENTS.
           WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
               AFTER ADVANCING 3 LINES.
           CLOSE IM-PERSONNEL-MASTER-FILE,
                 OR-PRINTER.
       0010-EXIT.
           EXIT.
       0020-READ-WRITE-RTN.
           READ IM-PERSONNEL-MASTER-FILE
               AT END
                   MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
                   GO TO 0020-EXIT.
           ADD IM01-NUMBER-OF-DEPENDENTS TO WS-TOTAL-DEPENDENTS.
           IF IM01-NUMBFR-OF-DEPENDENTS LESS THAN 4
               THEN
                   GO TO 0020-EXIT.
           IF WS-LINE-COUNTER GREATER THAN 15
               THEN
                   PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
           MOVE IM01-NAME TO WS-DL-NAME.
           MOVE IM01-RANK TO WS-DL-RANK.
           MOVE IM01-COMPANY TO WS-DL-COMPANY.
           ADD 1, IM01-NUMBER-OF-DEPENDENTS GIVING WS-DL-FAMILY-SIZE.
           WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
               AFTER ADVANCING 2 LINES.
           ADD 1 TO WS-LINE-COUNTER.
       0020-EXIT.
           EXIT.
       0030-HEADING-RTN.
           MOVE 0 TO WS-LINE-COUNTER.
           WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-1
               AFTER ADVANCING TOP-OF-PAGE.
           WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-2
               AFTER ADVANCING 3 LINES.
       0030-EXIT.
           EXIT.
```

WS-TL-NUMBER OF-DEPENDENTS

MOVE WS-TOTAL-DEPENDENTS

CHECK PROBLEM



No.____________

SCHOOLCRAFT, MICHIGAN __________ 19____ 74-827/724

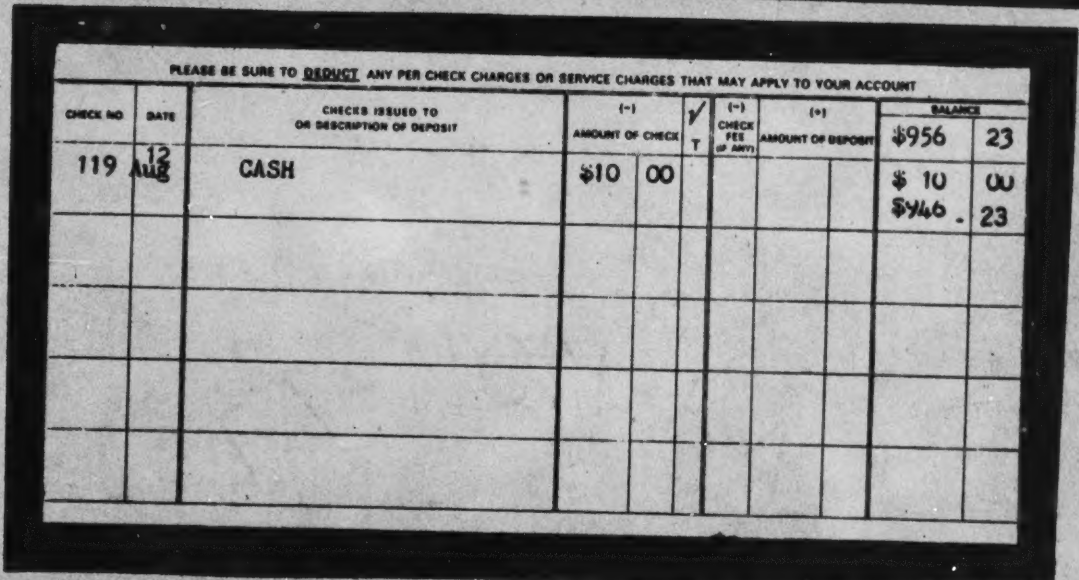
PAY TO THE ORDER OF ____________________ $________

______________________________ DOLLARS

SAMPLE

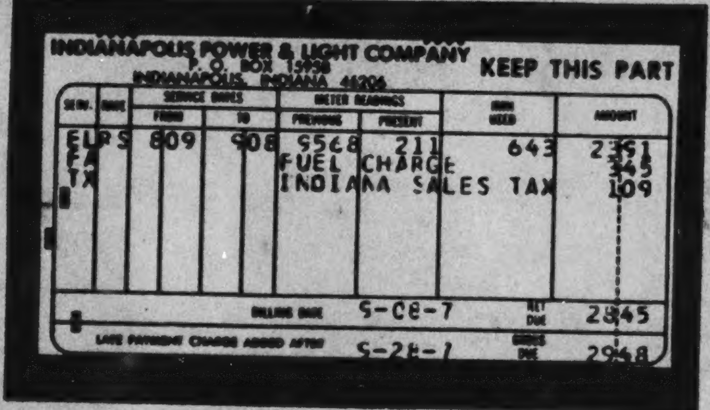
Kalamazoo County State Bank
SCHOOLCRAFT · MICHIGAN

⑆0724⑉0827⑆



PLEASE BE SURE TO DEDUCT ANY PER CHECK CHARGES OR SERVICE CHARGES THAT MAY APPLY TO YOUR ACCOUNT

CHECK NO	DATE	CHECKS ISSUED TO OR DESCRIPTION OF DEPOSIT	(−) AMOUNT OF CHECK		✓ T	(−) CHECK FEE (IF ANY)	(+) AMOUNT OF DEPOSIT		BALANCE	
									$956	23
119	12 Aug	CASH	$10	00					$ 10	00
									$946	23



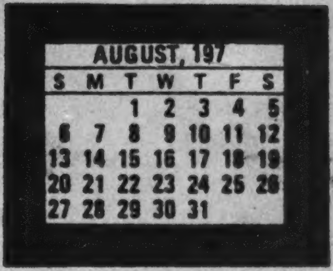
INDIANAPOLIS POWER & LIGHT COMPANY
P. O. BOX 15958
INDIANAPOLIS, INDIANA 46206

KEEP THIS PART

SERV.	RATE	SERVICE DATES FROM	SERVICE DATES TO	METER READINGS PREVIOUS	METER READINGS PRESENT	KWH USED	AMOUNT
EL	RS	809	908	9568	211	643	2351
FA				FUEL CHARGE			385
TX				INDIANA SALES TAX			109

BILLING DATE 5-08-7 NET DUE 2845

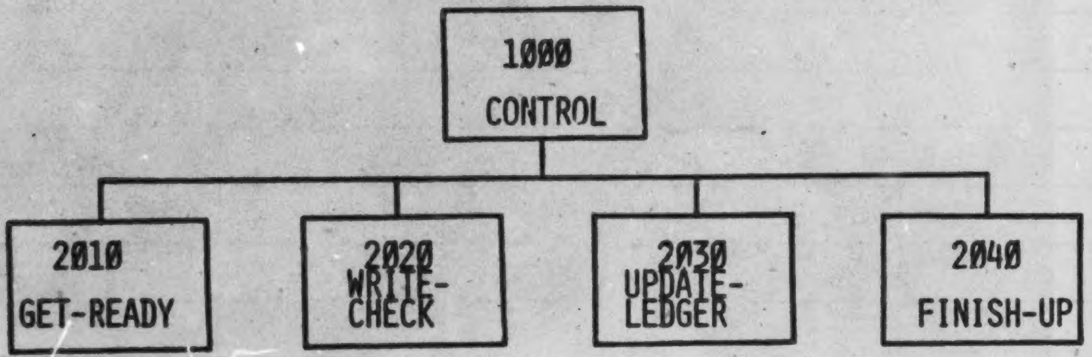
LATE PAYMENT CHARGE ADDED AFTER 5-28-7 GROSS DUE 2948



AUGUST, 197

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

CHECK PROBLEM HIERARCHY CHART


1000
CONTROL
2010
GET-READY
2020
WRITE-CHECK
2030
UPDATE-LEDGER
2040
FINISH-UP

PROGRAM NAME: CHECK PROBLEM

STUDENT NAME: KUCINICH

DATE: JAN 79

IPO

MODULE NUMBER: 1000

MODULE NAME: CONTROL

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. UNPAID-BILL	PERFORM 2010-GET-READY.	1. PAID-BILL
	PERFORM 2020-WRITE-CHECK.	
	PERFORM 2030-UPDATE-LEDGER.	
	PERFORM 2040-FINISH-UP.	
	STOP.	

"FOR INSTRUCTIONAL PURPOSES ONLY"

327

328

PROGRAM NAME: CHECK PROBLEM

STUDENT NAME: KUCINICH

DATE: JAN 79

IPO

MODULE NUMBER: 2010

MODULE NAME: GET-READY

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. UNPAID-BILL	OPEN UNPAID-BILL.	1. OPENED UNPAID-BILL
2. CHECK-BOOK	OPEN CHECK-BOOK.	2. OPENED CHECK-BOOK
3. CALANDER	EXTRACT TODAYS-DATE.	2.1 BLANK-CHECK
	EXIT.	2.2 CHECK-LEDGER
		3. TODAYS-DATE

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-01-23

PROGRAM NAME: CHECK PROBLEM
STUDENT NAME: KUCINICH
DATE: JAN 79

IPO

MODULE NUMBER: 2020
MODULE NAME: WRITE-CHECK
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. OPENED CHECK-BOOK	ADD 1 TO PREVIOUS-CK-NO.	1. WRITTEN-CHECK
1.1 BLANK-CHECK	MOVE PREVIOUS-CHK-NO TO CHECK-NO.	1.1 CHECK-NO
1.2 CHECK-LEDGER	MOVE BILL-PAYEE TO CHECK-PAYEE.	1.2 CHECK-DATE
1.2.1 PREVIOUS-	MOVE BILL-AMOUNT TO CHECK-AMT-NUMERIC.	1.3 CHECK-PAYEE
CHK-NO		
2. TODAYS-DATE	MOVE BILL-AMOUNT TO CHECK-AMT-ALPHA.	1.4 CHECK-AMOUNT
	MOVE TODAYS-DATE TO CHECK-DATE	
3. OPENED UNPAID-BILL	SIGN CHECK.	1.4.1 CHK-AMT-
		NUMERIC
3.1 BILL-PAYEE	EXIT.	1.4.2 CHK-AMT-ALPHA
3.2 BILL-AMOUNT		1.5 SIGNATURE

"FOR INSTRUCTIONAL PURPOSES ONLY"

4

329

PROGRAM NAME: CHECK PROBLEM
STUDENT NAME: KUCINICH
DATE: JAN 79

IPO

MODULE NUMBER: 2030
MODULE NAME: UPDATE-LEDGER
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CHECK	MOVE CHECK-NO TO LEDGER-CHK-NO.	1. UPDATED-LEDGER
1.1 CHECK-NO	MOVE CHECK-DATE TO LEDGER-DATE.	1.1 LEDGER-CHK-NO
1.2 CHECK-DATE	MOVE CHECK-PAYEE TO LEDGER-PAYEE.	1.2 LEDGER-DATE
1.3 CHECK-PAYEE	MOVE CHECK-AMT-NUMERIC TO LEDGER-AMT-	1.3 LEDGER-PAYEE
	OF-CHK.	1.4 LEDGER-AMT-OF-CHK
1.4 CHECK-AMT-	SUBTRACT CHK-AMT-NUMERIC FROM LEDGER-OLD-	1.5 LEDGER-NEW-BALANCE
NUMERIC	BALANCE.	
2. CHECK-LEDGER	MOVE LEDGER-OLD-BALANCE TO LEDGER-	
	NEW-BALANCE.	
2.1 LEDGER-OLD-BALANCE	EXIT.	

"FOR INSTRUCTIONAL PURPOSES ONLY"

PROGRAM NAME: CHECK PROBLEM
STUDENT NAME: KUCINICH
DATE: JAN 79

IPO

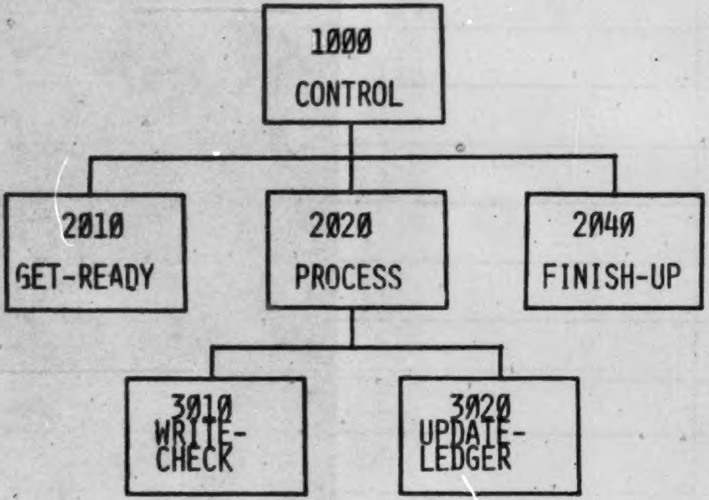
MODULE NUMBER: 2009
MODULE NAME: FINISH-UP
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. OPENED CHECK-BOOK	REMOVE WRITTEN CHECK FROM CHECK BOOK.	1. WRITTEN-CHECK
2. OPENED UNPAID-BILL	MARK BILL PAID.	2. CLOSED-CHECK-BOOK
	CLOSED CHECK-BOOK.	2.1 UPDATED-LEDGER
	EXIT.	3. PAID-BILL

"FOR INSTRUCTIONAL PURPOSES ONLY"

6

CHECK PROBLEM MODIFICATION HIERARCHY CHART


1000
CONTROL
2010
GET-READY
2020
PROCESS
2040
FINISH-UP
3010
WRITE-CHECK
3020
UPDATE-LEDGER

PROGRAM NAME: CHECK PROBLEM MODIFICATION
STUDENT NAME: KUCINICH
DATE: JAN 79

IPO

MODULE NUMBER: 1000
MODULE NAME: CONTROL
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. UNPAID-BILL(S)	PERFORM 2010-GET-READY.	1. PAID-BILL(S)
	PERFORM 2020-PROCESS UNTIL OUT OF BILLS.	
	PERFORM 2040-FINISH-UP.	
	STOP.	

2

333

IPO

PROGRAM NAME: CHECK PROBLEM MODIFICATION
STUDENT NAME: KUCINICH
DATE: JAN 79

MODULE NUMBER: 2020
MODULE NAME: PROCESS
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. UNPAID-BILL	PERFORM 3010-WRITE-CHECK.	1. PAID-BILL
	PERFORM 3020-UPDATE-LEDGER.	
	EXIT.	

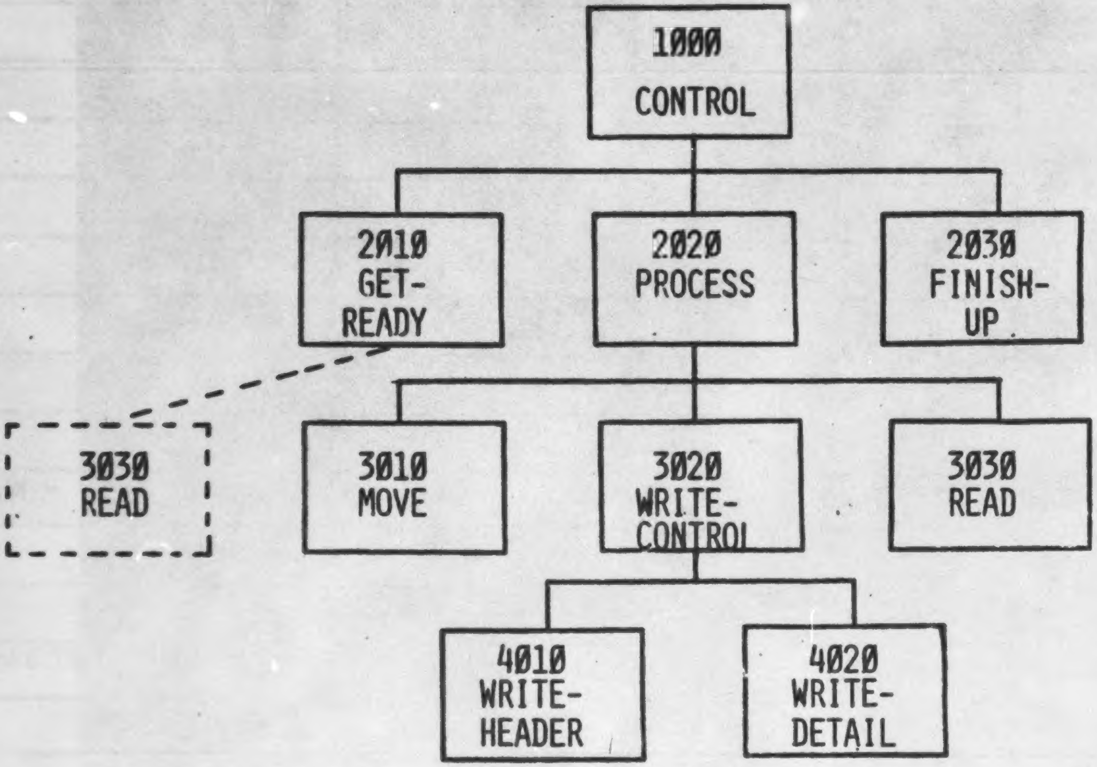
CHECK PROBLEM MODIFICATION

Now that you have solved the problem of writing a check and updating the check ledger to pay a bill, you will modify your solution to follow this process until you have payed all the bills.

You will assume that there is more than enough in the account to cover any and all bills that you must pay.

To accomplish this, you must redo your HIPO, and modify or create new IPOs.

PROBLEM SOLVING PE #1 HIERARCHY CHART


1000
CONTROL
2010
GET-
READY
2020
PROCESS
2030
FINISH-
UP
3030
READ
3010
MOVE
3020
WRITE-
CONTROL
3030
READ
4010
WRITE-
HEADER
4020
WRITE-
DETAIL

PROGRAM NAME: PROGRAM SOLVING PE #1
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 1000
MODULE NAME: CONTROL
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-FILE	PERFORM 2010-GET-READY.	1. PRINTED-REPORT
	PERFORM 2020-PROCESS UNTIL EOF-SW = 'ON'.	
	PERFORM 2030-FINISH-UP.	
EOF-SW = 'OFF' OR 'ON'	STOP RUN.	EOF-SW = 'OFF'

"FOR INSTRUCTIONAL PURPOSES ONLY"

2

337

61

360

PROGRAM NAME: PROBLEM SOLVING PE #1
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2010
MODULE NAME: GET-READY
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL-	OPEN INPUT PERSONNEL-MASTER-FILE.	1. OPENED PERSONNEL-
MASTER-FILE	OPEN OUTPUT PRINT-FILE.	MASTER-FILE
2. CLOSED PRINT-FILE	MOVE 'OFF' TO EOF-SW.	2. OPENED PRINT-FILE
	MOVE 'OFF' TO HEADER-SW.	
	PERFORM 3030-READ.	
EOF-SW	EXIT.	EOF-SW = 'OFF'
HEADER-SW		HEADER-SW = 'OFF'

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2020

MODULE NAME: PROCESS

PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
	PERFORM 3010-MOVE.	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
	EXIT.	

"FOR INSTRUCTIONAL PURPOSES ONLY"

4

339

340

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2030

MODULE NAME: FINISH-UP

PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. OPENED PERSONNEL-	CLOSE PERSONNEL-MASTER-FILE.	1. CLOSED PERSONNEL-
MASTER-FILE		MASTER-FILE
2. OPENED PRINT-FILE	CLOSE PRINT-FILE.	2. CLOSED PRINT-FILE
	EXIT.	

"FOR INSTRUCTIONAL PURPOSES ONLY"

5

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3010

MODULE NAME: MOVE

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	MOVE PMR-NAME TO DL-NAME.	1. FORMATED DETAIL-LINE
1.1 PMR-NAME	MOVE PMR-RANK TO DL-RANK.	1.1 DL-RANK
1.2 PMR-RANK	EXIT.	1.2 DL-NAME

"FOR INSTRUCTIONAL PURPOSES ONLY"

342

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3020

MODULE NAME: WRITE-CONTROL

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	IF HEADER-SW = 'OFF'	
	THEN PERFORM 4010-WRITE-HEADER	
	MOVE 'ON' TO HEADER-SW	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
HEADER-SW = 'OFF' OR 'ON'	EXIT.	HEADER-SW = 'OFF' OR 'ON'

"FOR INSTRUCTIONAL PURPOSES ONLY"

7

PROGRAM NAME: PROBLEM SOLVING PE #1
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 3030
MODULE NAME: READ
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	READ PERSONNEL-MASTER-FILE	1. PERSONNEL-MASTER-
	AT END	RECORD
	MOVE 'ON' TO EOF-SW.	
	EXIT.	
EOF-SW = 'OFF' OR 'ON'		EOF-SW = 'OFF' OR 'ON'

8

343

344

PROGRAM NAME: PROBLEM SOLVING PE #1
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 4010
MODULE NAME: WRITE-HEADER
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
	WRITE HEADER-RECORD AFTER ADVANCING TO	1. HEADER-RECORD
	TOP OF PAGE.	
	EXIT.	
1. HEADER-RECORD		
1.1 'RANK'		
1.2 'NAME'		

PROGRAM NAME: PROBLEM SOLVING PE #1
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 4020
MODULE NAME: WRITE-DETAIL
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. FORMATED-DETAIL-	WRITE DETAIL-RECORD AFTER ADVANCING 2	1. DETAIL-RECORD
	LINES.	
	EXIT.	

10

PROBLEM SOLVING PE #2

I. The records section here at Fort Harrison is not satisfied with the report from PE #1. They have asked for the following modifications.

Count the lines on each page and print only twenty (20) detail lines on each page.

Format the output report as follows;

```
                    FORT HARRISON PERSONNEL LISTING

    RANK                                 NAME

    XXX                          XXXXXXXXXXXXXXXXXXXX

    XXX                          XXXXXXXXXXXXXXXXXXXX
```

Print the heading lines at the top of each new page.

PROGRAM NAME: PROBLEM SOLVING PE #2
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2010
MODULE NAME: GET-READY
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL-MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE. OPEN OUTPUT PRINT-FILE.	1. OPENED PERSONNEL-MASTER-FILE
		2. OPENED PRINT-FILE
2. CLOSED PRINT-FILE	MOVE 'OFF' TO EOF-SW.	
	MOVE 99 TO LINE-CNTR.	
EOF-SW	PERFORM 3030-READ.	EOF-SW = 'OFF'
LINE-CNTR	EXIT.	LINE-CNTR = 99

1

348

PROGRAM NAME: PROBLEM SOLVING PE #2

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3020

MODULE NAME: WRITE-CONTROL

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	IF LINE-CNTR IS GREATER THAN 19	
	THEN PERFORM 4010-WRITE-HEADER	
	MOVE 0 TO LINE-CNTR	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
LINE-CNTR = (N)	ADD 1 TO LINE-CNTR.	LINE-CNTR = (N+1) OR 0
	EXIT.	

2

PROGRAM NAME: PROBLEM SOLVING PE #2

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 4010

MODULE NAME: WRITE-HEADER

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	WRITE HEADER-RECORD-1 AFTER ADVANCING TO	1. HEADER-RECORD-1
	TOP OF PAGE.	
	WRITE HEADER-RECORD AFTER ADVANCING 2	2. HEADER-RECORD
	LINES.	
	EXIT.	
1. HEADER-RECORD-1		
1.1 'FORT HARRISON		
PERSONNEL LISTING'		
2. HEADER-RECORD		
2.1 'RANK'		
2.2 'NAME'		

3

349

PROBLEM SOLVING PE #3

I INTRODUCTION.

The Battalion Headquarters at Fort Harrison has asked for a listing of all personnel assigned to or stationed here for accountability purposes. They would also like a count of all records listed.

II INPUT/OUTPUT.

INPUT: Input will consist of a personnel Master file, which contains records on all personnel stationed at or assigned to the Battalion.

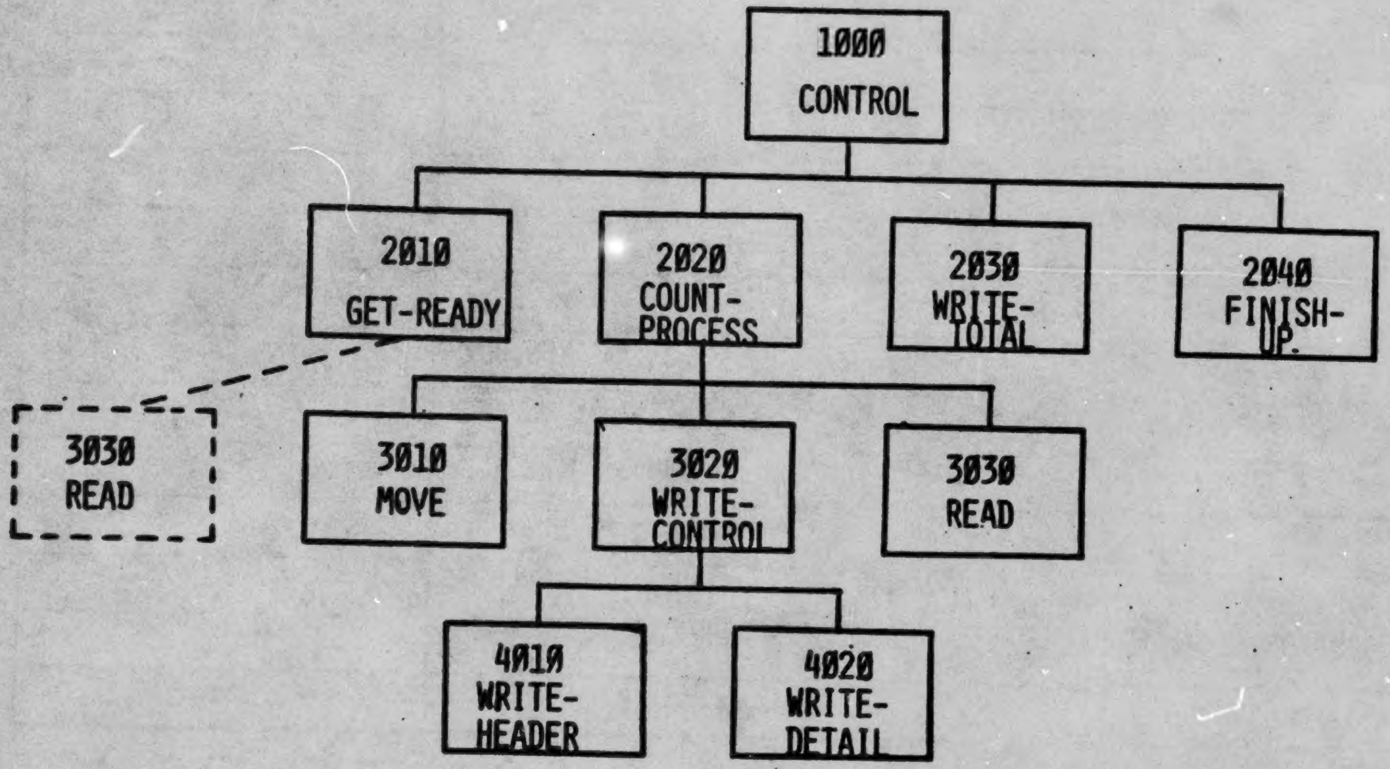
Personnel Master Records;
1. SOCIAL SECURITY ACCOUNT NUMBER
2. NAME
3. BATTALION
4. RANK
5. NUMBER OF DEPENDENTS

OUTPUT: Output will consist of a printed listing with a heading at the top of each page. Print a detail line for each record read, and a total line showing the number of records processed. Write 20 detail lines per page (double spaced) then start at the top of the next page with the header. The total line will be printed only once at the end of the report.

Format the output report as follows;

```
      SSN                  NAME                  RANK        NR DEPS

   999999999         xxxxxxxxxxxxxxxxxx          xxx            99

   999999999         xxxxxxxxxxxxxxxxxx          xxx            99

   999999999         xxxxxxxxxxxxxxxxxx          xxx            99

TOTAL RECORDS PRINTED 999
```

PROBLEM SOLVING PE #3 HIERARCHY CHART


1000
CONTROL
2010
GET-READY
2020
COUNT-PROCESS
2030
WRITE-TOTAL
2040
FINISH-UP
3030
READ
3010
MOVE
3020
WRITE-CONTROL
3030
READ
4010
WRITE-HEADER
4020
WRITE-DETAIL

IPO

PROGRAM NAME: PROBLEM SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

MODULE NUMBER: 1000
MODULE NAME: CONTROL
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	PERFORM 2010-GET-READY.	1. PRINTED-LISTING
FILE	PERFORM 2020-COUNT-PROCESS UNTIL	
	EOF-SW = 'ON'.	
	PERFORM 2030-WRITE-TOTAL.	
	PERFORM 2040-FINISH-UP.	
EOF-SW = 'OFF'	STOP RUN.	EOF-SW = 'ON'

"FOR INSTRUCTIONAL PURPOSES ONLY"

2

PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2010

MODULE NAME: GET-READY

PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL-MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE.	1. OPENED PERSONNEL-MASTER-FILE
2. CLOSED PRINT-FILE	OPEN OUTPUT PRINT-FILE.	2. OPENED PRINT-FILE
	MOVE 99 TO LINE-CNTR.	
	MOVE 'OFF' TO EOF-SW.	
	MOVE 0 TO PERS-CNTR.	
LINE-CNTR	PERFORM 3030-READ.	LINE-CNTR = 99
EOF-SW	EXIT.	EOF-SW = 'OFF'
PERS-CNTR		PERS-CNTR = 0

"FOR INSTRUCTIONAL PURPOSES ONLY"

PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2020

MODULE NAME: COUNT-PROCESS

PAGE 1 of 1

354

INPUT	PROCESS	OUTPUT
	ADD 1 TO PERS-CNTR.	
	PERFORM 3010-MOVE.	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
PERS-CNTR = (N)	EXIT.	PERS-CNTR = (N+1)

"FOR INSTRUCTIONAL PURPOSES ONLY"

4

PROGRAM NAME: PROBLEM SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2030
MODULE NAME: WRITE-TOTAL
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
	MOVE PERS-CNTR TO TOTAL-PERS.	1. TOTAL-LINE
	WRITE TOTAL-LINE AFTER ADVANCING 2 LINES.	
	EXIT.	
1. TOTAL-LINE		
1.1 'TOTAL RECORDS		
PRINTED'		
1.2 TOTAL-PERS		
PERS-CNTR = (N)		

121-013-1401-080-A "FOR INSTRUCTIONAL PURPOSES ONLY" IA-01-03-26 5

PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2040

MODULE NAME: FINISH-UP

PAGE 1 of 1

356

INPUT	PROCESS	OUTPUT
1. OPENED PERSONNEL-	CLOSE PERSONNEL-MASTER-FILE.	1. CLOSED PERSONNEL-
MASTER-FILE	CLOSE PRINT-FILE.	MASTER-FILE
2. OPENED PRINT-FILE	EXIT.	2. CLOSED PRINT-FILE

PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3010

MODULE NAME: MOVE

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	MOVE PMR-SSN TO DL-SSN.	1. FORMATED-DETAIL-LINE
RECORD	MOVE PMR-NAME TO DL-NAME	
1.1 PMR-SSN	MOVE PMR-RANK TO DL-RANK.	1.1 DL-SSN
1.2 PMR-NAME	MOVE PMR-NR-DEPS TO PMR-NR-DEPS.	1.2 DL-NAME
1.3 PMR-RANK	EXIT.	1.3 DL-RANK
1.4 PMR-NR-DEPS		1.4 DL-NR-DEPS

"FOR INSTRUCTIONAL PURPOSES ONLY"

7

PROGRAM NAME: PROBLEM SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 3020
MODULE NAME: WRITE-CONTROL
PAGE 1 of 1

358

INPUT	PROCESS	OUTPUT
	IF LINE-CNTR IS GREATER THAN 19 THEN	
	PERFORM 4010-WRITE-HEADER	
	MOVE 0 TO LINE-CNTR	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
	ADD 1 TO LINE-CNTR.	
LINE-CNTR = (N)	EXIT.	LINE-CNTR = (N+1) OR 0

PROGRAM NAME: PROBLEM-SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 3030
MODULE NAME: READ
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	READ PERSONNEL-MASTER-FILE	1. PERSONNEL-MASTER-
FILE	AT-END	RECORD
	MOVE 'ON' TO EOF-SW.	
	EXIT.	
EOF-SW = 'OFF'		EOF-SW = 'OFF' OR 'ON'

"FOR INSTRUCTIONAL PURPOSES ONLY"

PROGRAM NAME: PROBLEM SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 4010
MODULE NAME: WRITE-HEADER
PAGE 1 of 1

360

INPUT	PROCESS	OUTPUT
	WRITE HEADER-LINE AFTER ADVANCING TO TOP	1. HEADER-LINE
	OF PAGE	
	EXIT.	
1. HEADER-LINE		
1.1 'SSN'		
1.2 'NAME'		
1.3 'RANK'		
1.4 'NR DEPS'		

121-013-1401-080-A "FOR INSTRUCTIONAL PURPOSES ONLY" IA-01-03-26

IPO

PROGRAM NAME: PROBLEM SOLVING PE #3
STUDENT NAME: KUCINICH
DATE: FEB 79

MODULE NUMBER: 4020
MODULE NAME: WRITE-DETAIL
PAGE 1 **of** 1

INPUT	PROCESS	OUTPUT
1. FORMATED DETAIL-LINE	WRITE DETAIL-RECORD AFTER ADVANCING 2	1. DETAIL-RECORD
	LINES.	
	EXIT.	

"FOR INSTRUCTIONAL PURPOSES ONLY"

121-013-1401-000-A

TA-01-03-26

362

PROBLEM SOLVING PE #4

I INTRODUCTION.

The Battalion Headquarters was pleased with the report produced from PE #3 but feel they could better use the report with the following modifications.

1. Print only those records of personnel above the rank of PV1.

2. Print an additional total line two lines below the one now being generated which contains a count of the number of PV1's on the Personnel Master File.

Format of second total line is as follows;

TOTAL NUMBER OF PV1 999

PROGRAM NAME: PROBLEM SOLVING PE #4
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2030
MODULE NAME: WRITE-TOTAL
PAGE ____ **of** ____

INPUT	PROCESS	OUTPUT
	MOVE PERS-CNTR TO TOTAL-PERS.	1. TOTAL-LINE
	WRITE TOTAL-LINE AFTER ADVANCING 2 LINES.	2. TOTAL-LINE-2
	MOVE PV1-CNTR TO TOTAL-PV1.	
1. TOTAL-LINE	WRITE TOTAL-LINE-2 AFTER ADVANCING 2	
1.1 'TOTAL RECORDS	LINES.	
PRINTED'		
1.2 TOTAL-PERS		
2. TOTAL-LINE-2		
2.1 'TOTAL NUMBER OF		
PV1'		
2.2 TOTAL-PV1		
PERS-CNTR = (N)		
PV1-CNTR = (N)		

363

410

409

364

PROGRAM NAME: PROBLEM SOLVING PE #4
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2020
MODULE NAME: COUNT-PROCESS
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	IF PMR-RANK = 'PV1'	
RECORD	THEN ADD 1 TO PV1-CNTR	
1.1 PMR-RANK	ELSE ADD 1 TO PERS-CNTR	
	PERFORM 3010-MOVE	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
PERS-CNTR = (N)	EXIT.	PERS-CNTR = (N) OR (N+1)
PV1-CNTR = (N)		PV1-CNTR = (N) OR (N+1)

2

411

412

PROGRAM NAME: PROBLEM SOLVING PE #4
STUDENT NAME: KUCINICH
DATE: FEB 79

IPO

MODULE NUMBER: 2010
MODULE NAME: GET-READY
PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL-MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE.	1. OPENED PERSONNEL-MASTER-FILE
2. CLOSED PRINT-FILE	OPEN OUTPUT PRINT-FILE.	2. OPENED PRINT-FILE
	MOVE 99 TO LINE-CNTR.	
	MOVE 'OFF' TO EOF-SW.	
	MOVE 0 TO PERS-CNTR.	
LINE-CNTR	MOVE 0 TO PV1-CNTR.	LINE-CNTR = 99
EOF-SW	PERFORM 3030-READ.	EOF-SW = 'OFF'
PERS-CNTR	EXIT.	PERS-CNTR = 0
PV1-CNTR		PV1-CNTR = 0

"FOR INSTRUCTIONAL PURPOSES ONLY"

3

SORT EXERCISES

1. Code the jobstream to sort a file. I/O specifications follow (Use 1 sort work area):

	Input	Output
Media:	Tape	Tape
Rec Length:	80/1600	80/1600
File ID:	PERS-TX	SORTED-PERS-TX
Volume of Data	500 records	500 records
Device:	TP0	TP1
Control Field	pos 8-15 (character data)	

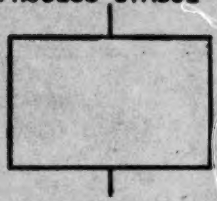
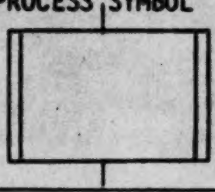
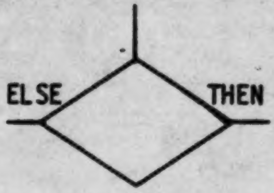
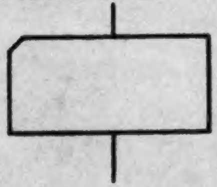
2. Assume that you have 3 sorted files in the format of problem 1. Code the jobstream to merge them.

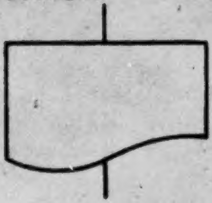
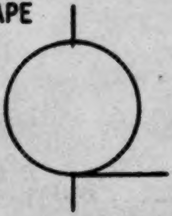
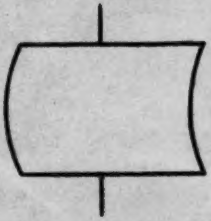
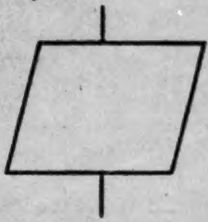
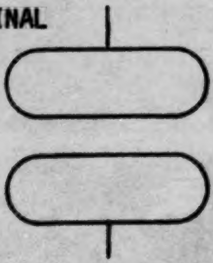
3. Code the jobstream to sort the following files (Use 2 sort work extents):

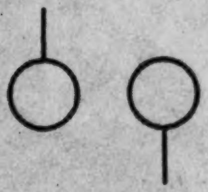
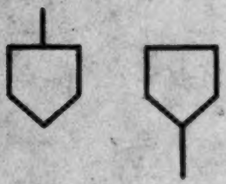
	Input1	Input2	Output
Media:	Tape	Tape	Disk
Rec length/Blksize:	100/1000	100/1000	100/7000
File ID:	LOG-MSTR	LOG-TEMP	SORTED-LOG-MSTR
Volume of data:	20000 records	30000 records	50000 records
Device:	TP1	TP0	DK4
Volume Serial Nr:	N/A	N/A	LOG PAK
Retention:	N/A	N/A	3 months
Control fields:			
MAJOR:	8-20, char		
INT 1:	29-30, char		
INT 2:	24, char		
MINOR:	30-36, char		

STRUCTURED FLOWCHARTING SYMBOLS

The flowcharting symbols defined in this handout are to be used in conjunction with the Verbs defined in the COBOL SOURCE DESIGN LANGUAGE handout. This handout does not contain all of symbols that can be used in structured flowcharting but is limited to those symbols that will be used in the problem solving block of instruction.

SYMBOL	USE
PROCESS SYMBOL 	The process symbol may be used to represent the ADD, CLOSE, OPEN, MOVE or NEXT SENTENCE. This symbol requires one entry and one exit.
PREDEFINED PROCESS SYMBOL 	The predefined process symbol may be used only to represent a PERFORM. This symbol requires one entry and one exit.
DECISION 	The decision symbol may be used only to represent the IF-THEN-ELSE. This symbol requires one entry and two exits, one THEN exit and one ELSE exit.
PUNCHED CARD 	The punched card symbol may be used to represent a READ for an input punched card file or a WRITE of a punched card record. When used for a read it will have one entry and two exits. When used for a write it will have one entry and one exit.

SYMBOL	USE
DOCUMENT 	The document symbol may be used only to represent a WRITE to a printer. This symbol requires one entry and one exit.
MAGNETIC TAPE 	The magnetic tape symbol is used in the same way as the punched card symbol except where the storage media is magnetic tape.
ONLINE STORAGE 	The online storage symbol is used in the same way as the punched card symbol except where the storage media is a disk.
INPUT/OUTPUT 	The input/output symbol is used in the same way as the punched card symbol except where the storage media is unknown.
TERMINAL 	The terminal symbol is used to show the beginning and ending point of each routine. It will contain the module number and name or the word EXIT or STOP RUN. This symbol requires either one entry or one exit but not both.

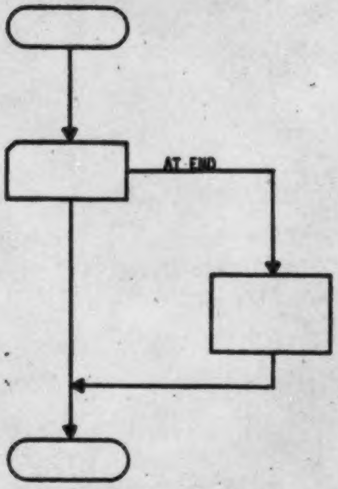
SYMBOL	USE
ON PAGE CONNECTOR 	The on page connector symbol is used to connect two parts of the same routine which are drawn separately on the same sheet of paper.
OFF PAGE CONNECTOR 	The off page connector symbol is used to connect two parts of the same routine which are drawn separately on separate sheets of paper.

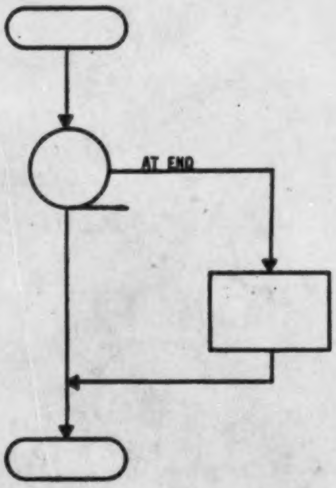
When using the PUNCH CARD, MAGNETIC TAPE, ONLINE STORAGE or INPUT/OUTPUT symbol for a READ, you will always show the AT END associated with it as follows: (See page 4)

For an illistration of the use of an IF-THEN-ELSE and a compound IF-THEN-ELSE see page 5.

IBM Flowcharting Worksheet FOR INSTRUCTIONAL PURPOSES ONLY

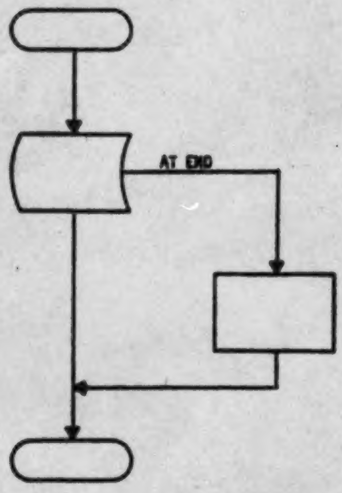
Programmer: Kucinich Program No.: Date: Jan 79 Page: 4
Chart ID: Chart Name: READ routine structured flowcharts Program Name:

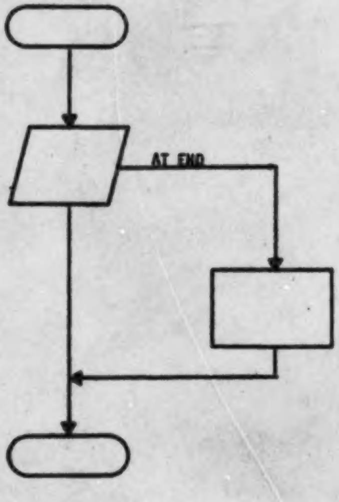




Fold under at dotted line.

Fold under at dotted line.

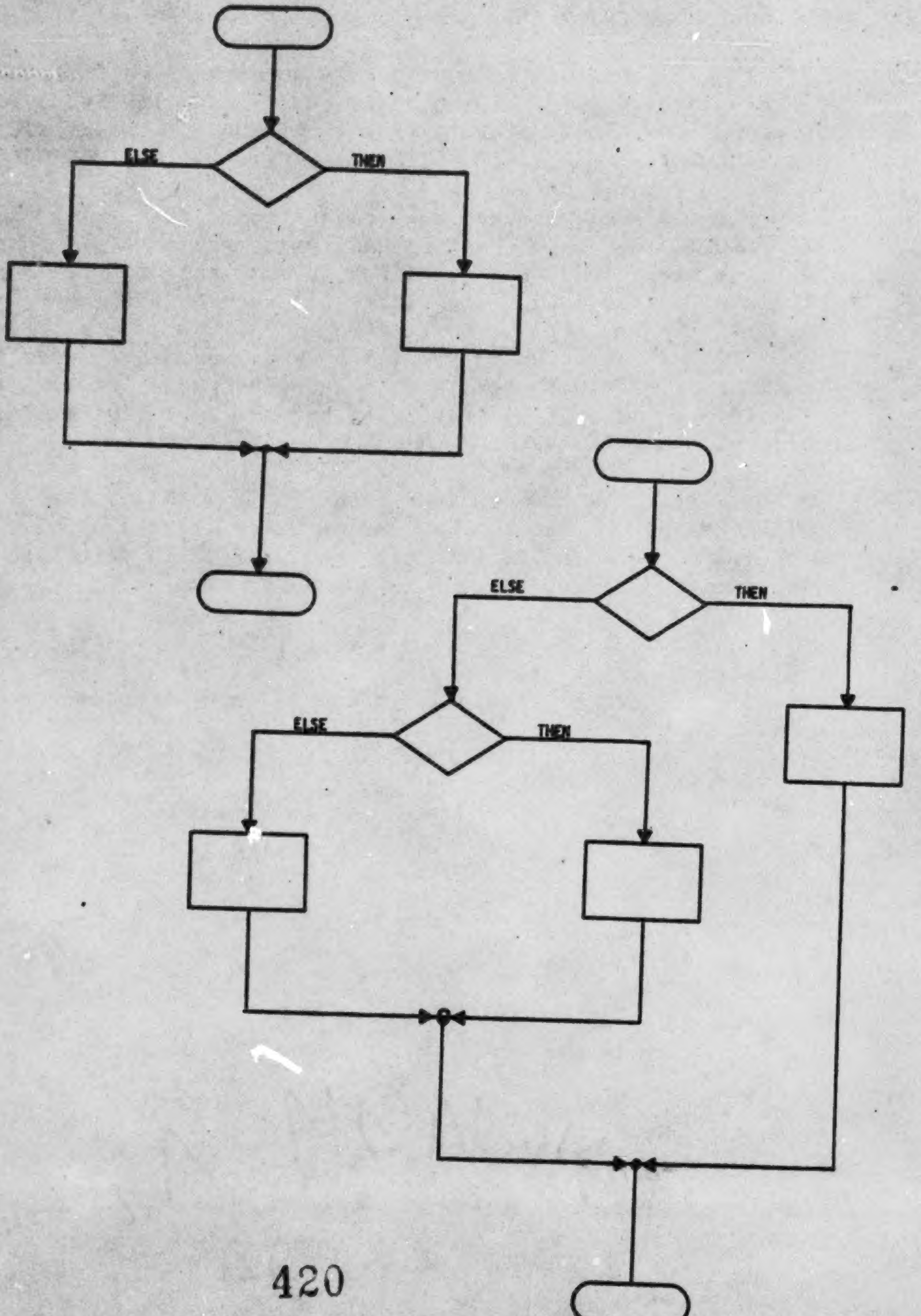




IBM Flowcharting Worksheet FOR INSTRUCTIONAL PURPOSES ONLY

Programmer: Kucinich Program No.: Date: Jan 79 Page: 5

Chart ID: Chart Name: IF-THEN-ELSE structured flowchart Program Name:



Fold under at dotted line.

Fold under at dotted line.

UTILITY EXERCISES

1. Code the jobstream to copy a deck of cards to tape. Columns 21 and 30 must be interchanged on the output tape, additionally columns 71-80 must be switched with columns 1-10. All other columns remain the same. The output tape must have 20 records per block and should be unloaded at EOJ.

2. Code the jobstream to list a file on disk. The file is on tracks 80-200 on pack DATA19. The records on disk are 100 bytes long and blocked 20 records per block. The listing should be single spaced. A 2314 disk will be used. The file-ID of the disk file is 'MOD WORKOUT'.

3. Write the jobstream to punch cards from a disk file. The file is called 'PAY RECORDS' and is on pack PAYPAC (omit extent info). The records are 120 bytes blocked by 20. The first 40 and last 40 bytes make up the 80 columns of output data. The first 1500 records should be bypassed.

4. Write the jobstream to print the cards punched in problem 3. All 80 bytes are to be printed with the output having 5 spaces between each group of 40 bytes (i.e.: 40 bytes 5 spaces, 40 bytes). The listing should be double spaced with page numbers.

5. Write the jobstream to load a tape file to disk. The input file-ID is 'A01ABC TRANS' and the record length is 100 blocked by 30. The output file is stored on pack DLOG01 with extents and file-ID determined by the programmer. Output retention is one month.

6. Write the jobstream to copy a tape file. Input tape is 'PAY FILE' created on 6 Jan and is 90 bytes per record blocked by 10. Output should be blocked by 30 and be retained 3 months. Input file is generation 5, output is generation 6.

UTILITIES P.E.

The PSNCO of your unit has brought you a deck of cards that he wants listed. The format for these cards is attached. The cards represent all enlisted personnel in the battalion. He does not know the current sequence of these cards, but he requires the output to be printed in sequence according to the print record layout below. Code the jobstream that will sort the cards and print them according to the customer's desires.

NOTES:

1. The SORT program will not read cards or print so two utility programs are required; one to transfer the card data to disk and another to print the sorted file.

2. Use 1 sort work file on disk.

3. Use ADAS JCL for all disk files (Volume serial no = SYSWRK).

Field	Sequence	Card	Print
Social Security Number		1-9	N/A
Name	Minor; A	11-30	21-40
MOS		32-38	42-48
Grade	Int; D	40-42	50-52
Company	Major; D	44	14
Date of Rank		46-51	56-61

374

DISK OPERATING SYSTEM
INTERFILE TRANSFER AND MANIPULATOR PROGRAM
FOR AID IN TESTING AND OPERATIONS
"DITTO"

CSD.SFW.DITTO

01-01-06

TABLE OF CONTENTS

I.	Title Page	1
II.	Table of Contents	2
III.	Program Abstract	3
IV.	Description of Functions	4
V.	Control Card Operation	10
VI.	Console Operating Instructions	12
VII.	Multiprogramming Considerations	13
VIII.	Table of Function Codes	15
IX.	Operating Suggestions	17
X.	Implementation	19
XI.	Tape and Deck Key	20
XII.	Sample Programs	21

Ditto is used throughout this documentation in lieu of the program title, "DOS Interfile Transfer and Manipulator Program For Aid In Testing and Operations."

376

PROGRAM ABSTRACT

DOS Interfile Transfer And Manipulator Program For Aid In Testing And Operations - (DITTO), is a generalized program which operates in a DOS environment. Thirty-three (33) functions are available to support Unit Record, Tape and 2311 and 2314 Disk. Functions available in addition to normal tape, disk and card functions are: Tape and Disk Record Scan, Disk and Tape Record Alteration, Disk ID Volume Number Change, Initialize Tape, Deblocking of Tape and Disk Records When Printing, and User Tape Error Handling.

Operating Characteristics Are:

1. The program is self-relocating to allow operating in any partition. Tape and Disk record size is limited only by the amount of core available at execution time.

2. Operations may be performed entirely from the console or function card statements may be job-streamed for remote programmer use.

3. All console communication is in terms of physical hardware I/O addresses, eliminating the need for knowledge of current logical assignments. Tape density and mode settings may be entered via the console.

4. DITTO functions in a DASD file protected environment without volume label statements. Printed output reflects the physical and logical addresses and file characteristics (density, etc.) of the input file.

5. Minimum requirements: System/360 DOS System with a 1403 Printer and a minimum 14K problem program area. The program is written in Assembler language.

DESCRIPTION OF FUNCTIONS

Disk Print

2311 or 2314 Disk to Printer in character and vertical format within specified limits. Disk records are "separated" into key and data format and printed with their associated cylinder, head, and record addresses. When a defective track is detected, the assigned alternate track with its respective data is printed. This provides a sequential listing of the disk file in "logical" sequence.

Two Disk to Printer formats are available: Disk Dump Unblocked; and Disk Dump with Reblocking. The Unblocked format lists the actual records as they exist on the file. The Reblock function allows the user to submit a logical record size. This size parameter should be equivalent to the actual logical record length only, and should not include separate key length. Imbedded keys should, however, be included. The Reblock function lists the record key and then deblocks the data portion of the record into logical lengths. This provides a more easily used listing of large physical record files.

The Reblock function is primarily intended for use on sequential or random files. Because of the different formats present on index sequential files, reblocking may not be meaningful for overflow areas or cylinder and track indices.

Disk Record Scan

Disk Record Scan provides the capability of scanning a disk file and recording the locations of all records which match a given scan argument. Three (3) scan argument types are available.

"Scan on Key" allows the user to scan for a match on any portion of the disk record key fields. A 1 to 35 position scan argument, and its associated starting position within the key field, are entered via the console. Beginning and ending disk limits are also entered. DITTO will scan the specified disk area and log all "hits" in the form of cylinder, head and record numbers on SYSLOG.

"Scan on Data" provides a similar capability for the data portion of disk records. A logical record length must be specified to allow internal deblocking for scanning blocked files.

"Scan on EOF" will log all End of File record locations. Data entry for "EOF" is not required.

Disk Record Load

This function will alter the contents of the key and/or data portion of an existing 2311 or 2314 Disk Record. After the required disk unit and record addresses have been entered, the record will be retrieved and "separated" into key and data format and printed on SYSLST.

At this point, DITTO will request the number of bytes to be changed and the field type (Key or Data portion of the record) to be altered. After specifying these parameters, a starting position within the field will be requested. This starting position is based on using the first position of the key or data as position "0001". The user can verify this position by referencing the printout of the record and its associated scale on SYSLST. Data may be entered via the console in character (1 character per byte of change), or hexidecimal (2 characters per byte of change). Valid hexidecimal "characters" are 0-9 and A through F.

After all alterations to the record in core have been made, the altered record will again be printed on SYSLST and a message issued "ARE CHANGES COMPLETE - Y OR N." A response of "N" (NO) will allow the user to re-enter the alter routine. A response of "Y" (YES) will rewrite the record onto disk.

Write Disk End-of-File Record

This function will retrieve any existing 2311 or 2314 Disk Record and rewrite the record as an end-of-file record (key and data length of zero). The user must supply the cylinder, head, and record address of the record to be altered.

Split Cylinder Disk Organizations

Split cylinder data file organizations are supported by DITTO for the Disk Dump Unblocked (DDU), Disk Dump Reblocked (DDR), and the Disk Record Scan (DRS) functions. Split cylinder operations are denoted by replacing the first D in the above function codes with an S. For these split cylinder operations, the beginning and ending head addresses will determine the lower and upper head boundaries for the function.

Disk Identification Change

This function will print on SYSLOG the existing disk volume serial number and allow the user to enter on SYSLOG a new volume serial number.

Initialize Tape

Initialize tape using the standard DOS format with a tape volume serial number entered via the console. Existing tape labels or data on the output tape is not checked before creating the new label set.

Tape To Tape

Tape-to-tape copy including tape marks. This function will copy from 1 to 99 tape files per volume as requested via the console. One file will be copied per control card if console operation is not being used. The first file copied may or may not have a leading tape mark. A leading tape mark does not itself constitute a separate file. Block count statistics will be printed on SYSLST for control card operation, and on SYSLOG for console operation, for each tape file copied.

Tape Record Load

This function is used to copy and selectively alter existing tape records. The user need not know the exact location (block #) of the input tape record to be altered.

The first portion of the Tape Record Load function locates the desired record. The user enters an estimated block location and the parameter "F" to denote copying of tape in a forward position. When the desired record is found,it is printed on SYSLST and an option given to alter the record. All input records prior to the record to be altered have been copied onto the output tape. If the printed record is not the desired record, the user can again enter the # of blocks and direction to reposition the input and output tapes. If the user has gone past the desired record, a parameter of "B" should be entered with a block count. This will cause both input and output tapes to be backspaced the required number of blocks. This count includes the block just printed. When the desired record is located, the user may alter the length and/or data content in a manner similar to the Disk Record Load function. After all alterations are completed, replies of "9999" and "F" (9999 blocks to be copies in a forward direction) will usually be sufficient to finish copying the tape file.

Tape Record Scan

This function allows the user to specify a logical record size and a 1 to 35 position scan argument for an input tape. DITTO will read and internally deblock input records until the desired record is located. At that time the entire physical record with its associated block number will be printed and the operation terminated. The tape will not be rewound.

Tape Print

Four printing formats are available for input tapes. Tape data may be printed in character format only, or in character and vertical hex format. Physical tape records may be printed as they exist (unblocked) or may be deblocked into logical records. Block and record counts are printed with the associated records.

For control card operation, an optional NBLKS=nnnn parameter may be specified. "nnnn" specifies the maximum number of physical tape record blocks to be printed. If "nnnn" is greater than the existing number of blocks in the tape file, only the existing tape file will be printed. A leading tape mark will be automatically skipped.

Print SYSLST Tapes

Tape to printer unblocked using the first position of the tape record as the carriage control character. Two carriage control formats are available. Type A forms control (TFA) uses the carriage control character as the CCW operation code. DITTO performs its printing using Type A. Type D forms control (TFD) is used by DOS for compiler listings, etc.

Tape Control

Eight tape control functions are available either by control cards or console communications. These control functions are: write tape mark; rewind tape; rewind and unload tape; forward space or backspace file; forward space or backspace records; and erase record. The erase record will remove approximately 4" of tape data.

Card To Tape

Card image to tape with or without reblocking. Job control cards, including /* and /& cards, may be interspersed in the input data (see Card Input Assignment). A tape mark will be written after the last card block on the tape file.

Tape To Card

Tape to card for blocked or unblocked 80 character logical records. A leading tape mark will be bypassed if present. If the input tape record is not a multiple of 80 bytes, the function will terminate.

Card To Printer

Cards may be listed on SYSLST in either character or character and vertical hex format. Job Control cards including /* and /& statements may be interspersed in the input data (see Card Input Assignment).

Card To Card

Two card to punch options are available. CCU is an 80/80 reproduction of the input cards. CCS is a card-to-card copy with sequence numbers and deck identification name added. Three decktypes are available.

- DECKTYPE=COB is used to sequence and name COBOL source decks. Sequence numbers will be placed in CC 1-6 and a 0-8 position deckname placed in CC 73-80.
- DECKTYPE=RPG is used for RPG source decks. Sequence numbers will be placed in CC 1-5 and a 0-6 position deckname placed in CC 75-80.
- DECKTYPE=BAL is used for Assembler and FORTRAN Source decks, Job Control decks, etc. Sequence numbers will be placed in CC 77-80 and a 0-4 position deckname placed in CC 73-76.

Decknames will be left-justified and padded with blanks if less than the required length. /* and /& cards may be interspersed in the input data (see Card Input Assignment).

End Of Job

All functions, whether control card or console initiated, must issue the function code "EOJ" to return control to Job control. For control card operations, this "EOJ" card must be the last function requested.

Tape Errors

All tape errors will initially be handled by the DOS supervisor. For input tape errors that are uncorrectable, DITTO will transfer control to a console routine. This routine will display the record as read, and allow the operator to accept, bypass, or alter the data and/or length of the tape record before returning control to the original operation. This option is not available for remote programmer control card operation.

Card Input Assignment

If the user does not desire to include /* and/or /& cards as a part of his input data, normal DOS procedures should be followed. For control card operations, the end of the input deck is signaled with a /* card. This will denote end-of-file for the particular operation and is consistent with DOS operating procedures. The /* card may be followed with additional DITTO operations in a similar manner. Operations which do not require data card input should not contain a /* card. The last DITTO operation must be an EOJ card and should be followed by a /* card if other DOS job steps follow, or by a /& card if a new job is to be initiated by Job Control. Console operations for data card input may be performed in a similar manner. For console operations which do not have a stacked job input stream, /* and /& cards may be omitted. End-of-file on card input will be denoted by physical end-of-file on the input device.

The DOS Supervisor monitors all cards read by the problem program from SYSIPT or SYSRDR. If the problem program attempts to read past a /& statement using either of these system logical units, the Supervisor cancels the job. The Supervisor does not, however, monitor data from a programmer logical unit. To read /& statements, DITTO must, therefore, read card input data from a programmer logical unit.

The inclusion of /* and /& cards in the input is available under console operation only. To initiate a card input operation with /* and/or /& cards in the input data, the operator must substitute an "&" for the first "C" in the function code (i.e. CCS becomes &CS). A blank card in the input stream denotes end-of-file.

CONTROL CARD OPERATION

If the user wishes to remove the decision requirements from the console operator, he may prepare control cards and submit the task in a normal Job Stream environment.

To denote control card operation the user must submit between the Job card and Exec card a // UPSI 1 card. DITTO will test the communications region to determine whether control card or console operation is desired. Depressing the interrupt key at any time overrides the control card option.

The required control card information can be contained on one card. Each control card is treated as a new operation. The control card format is as follows:

CC 1 -7	$$DITTO
CC 10 -12	Function Code (see table)
CC 16	Parameter 1,......,Parameter N

Parameters are in standard key word format. Each parameter must be separated with a comma with no embedded blanks. A blank stops the card scan. Lower case letters denote user implied information. Parenthesis () denote optional parameters. Refer to the "Parameter Requirements" for a listing of parameters associated with each function.

Parameter	Description
INPUT=SYSnnn	Logical input device
OUTPUT=SYSnnn	Logical output device
BEGIN=ccchh	Lower disk extent
END=ccchh	Upper disk extent
NBLKS=nnnn	Number of tape blocks
RECSIZE=nnnnn	Logical record size
BLKFACTOR=nnn	Output blocking factor (CTR)
DECKTYPE=xxx	CCS decktype
DECKNAME=x....x	CCS deckname (0-8) characters. Decknames which are less in length than required will be left

justified and padded with blanks.

PARAMETER REQUIREMENTS

CC 1-7	CC 10-12	C 16
$$DITTO	DDU	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh
$$DITTO	DDR	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh,RECSIZE=nnnnn
$$DITTO	SDU	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh
$$DITTO	SDR	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh,RECSIZE=nnnnn
$$DITTO	TPU	INPUT=SYSnnn(,NBLKS=nnnn)
$$DITTO	TPR	INPUT=SYSnnn,RECSIZE=nnnnn(,NBLKS=nnnn)
$$DITTO	THU	INPUT=SYSnnn(,NBLKS=nnnn)
$$DITTO	THR	INPUT=SYSnnn,RECSIZE=nnnnn(,NBLKS=nnnn)
$$DITTO	TFA	INPUT=SYSnnn
$$DITTO	TFD	INPUT=SYSnnn
$$DITTO	TCR	INPUT=SYSnnn
$$DITTO	CTU	OUTPUT=SYSnnn
$$DITTO	CTR	OUTPUT=SYSnnn,BLKFACTOR=nnn
$$DITTO	TTU	INPUT=SYSnnn,OUTPUT=SYSnnn
$$DITTO	WTM	OUTPUT=SYSnnn
$$DITTO	REW	OUTPUT=SYSnnn
$$DITTO	RUN	OUTPUT=SYSnnn
$$DITTO	FSF	OUTPUT=SYSnnn
$$DITTO	BSF	OUTPUT=SYSnnn
$$DITTO	FSR	OUTPUT=SYSnnn,NBLKS=nnnn
$$DITTO	BSR	OUTPUT=SYSnnn,NBLKS=nnnn
$$DITTO	CPU	
$$DITTO	CHU	

CC 1-7	CC 10-12	C 16
$$DITTO	CCU	
$$DITTO	CCS	DECKTYPE=xxx,DECKNAME=xx...x
$$DITTO	EOJ	

CONSOLE OPERATIONS

All operator communication may be in upper or lower case. The cancel key is also implemented allowing retyping of replies in error. The end of message may be denoted by the "EOB" key or by repeatedly depressing the space bar.

Depressing the interrupt key for background operation will cause the existing function to be terminated and a new operation request displayed on the console. (See Multiprogramming Considerations for interrupting foreground operation.)

Tape address replies are entered in the form of "CUUMM" where C=channel, UU=unit address, and MM=the density and mode setting. For nine-track single density tapes, blanks or zeros may be substituted for C0, or the MM parameter may be omitted.

If the operator desires the inclusion of /* and/or /& cards in the input data for card input functions (i.e. loading Job Streams on tape, etc.), he must replace the first "C" of the function code with a "&". For these operations, a blank card denotes end-of-file. If the operator desires to cancel a program with card data input, he should interrupt DITTO and enter "CCL" on the console. This will cause the input stream to be flushed up to and including the end-of-file card.

DITTO may also be executed in the background entirely from the console by entering:

```
// JOB Anyname
// EXEC DITTO
```

After completing the desired operations, including the function "EOJ", a "READY FOR COMMUNICATIONS" message will be printed by Job Control. Reply:

```
/&
```

MULTIPROGRAMMING CONSIDERATIONS

Interrupting Foreground Operations

To interrupt DITTO when executing in a background partition, the external interrupt key on the CPU is depressed. This will cause the existing DITTO function to be terminated and a console message issued requesting a new function.

The interrupt DITTO when executing in a foreground partition, the following procedures should be used:

1. Depress Console Request Key.

2. Message "AR READY FOR COMMUNICATIONS" will be displayed.

3. Enter "MSG Fn", where n=1 or 2 and correspond to the partition being interrupted.

4. Enter EOB (Alt. coding 5).

The existing foreground DITTO function will be terminated and a console message issued requesting a new function.

Batch Job Foreground Operation (DOS Version III Only)

Requirements for Batch Job Foreground operation are identical to those of background operations. Control card operations may be interchanged in any batch partition. System units (SYSLST, SYSIPT, SYSPCH) must be assigned if used in any of the DITTO functions.

Foreground Initiator Operation

Foreground Initiator Operation of DITTO is available in the console mode of communication only. Control card operation, due to the lack of an active communications region for the partition, is not available. Due to the lack of system units (SYSIPT, SYSPCH, SYSLST) in the foreground for DOS Version II, SYS001,SYS002 and SYS003 should be assigned for the above devices respectively. These system unit restrictions do not apply to DOS Version III users.

System Unit Assignments

DITTO will determine at load time the device types assigned to SYSLST and SYSPCH. If SYSLST is assigned to tape, 133 character records will be written with the first character being the Type A carriage control character. If SYSPCH is assigned to tape, 80 character records will be written. At EOJ, a tape mark will be written and the tape backspaced for the appropriate System Unit if data has been written.

If SYSPCH is assigned to a unit record device, the appropriate write command for either a 2500 series, 1442 N1, or 1442 N2 punch will be used based on the user's supervisor. No DITTO modifications should be made for these devices.

386

FUNCTION CODES

Code	Alternate	Description
CCU	CC	Card to Card
CCS	--	Card to Card with sequence numbers and deck name identification
CPU	CP	Card to Printer in character format
CHU	CH	Card to Printer in character and vertical hexidecimal format
CTU	CT	Card to Tape Unblocked
CTR	--	Card to Tape Reblocked
TCR	TC	Tape to Card unblocked or blocked
TPU	TP	Tape to Printer unblocked in character format
TPR	--	Tape to Printer reblocked in character format
THU	TH	Tape to Printer unblocked in character and vertical hexidecimal format
THR	--	Tape to Printer reblocked in character and vertical hexidecimal format
TTU	TT	Tape to Tape (1 to 99 tape files)
*INT	--	Initialize Tape - DOS standard format
*TRL	--	Tape Record Load. Copy tape to tape with altering of length and/or data of specific records
*TRS	--	Tape Record Scan. Search tape for logical record using 1-35 position scan argument
TFA	--	Tape to Printer using Type A forms control
TFD	--	Tape to Printer using Type D forms control
WTM	--	Write Tape Mark
REW	--	Rewind Tape
RUN	--	Rewind and Unload Tape
FSF	--	Forward Space tape file

CODE	ALTERNATE	DESCRIPTION
BSF	--	Backspace tape file
FSR	--	Forward Space tape record
LSR	--	Back space tape record
ERG	--	Erase Record Gap
DID	—	Disk Identification change (change volume serial number)
DRL	--	Disk Record Load. Alter key and/or data portion of any disk record.
DDU	DD	Disk Dump. Disk to printer in character and vertical hexidecimal format unblocked.
DDR	--	Disk Dump with reblocking of data records into logical record length.
DRS	--	Disk Record Scan. Scan disk for matching key or data argument or for EOF.
SDU	SD	Disk Dump for split cylinder files
SDR	--	Disk Dump reblocked for split cylinder files
SRS	--	Disk Record Scan for split cylinder files
EOF	--	Write disk end-of-file record
CCL	--	Cancel card input function
EOJ	--	End of Job

* Denotes functions not available with control card operation.

OPERATING SUGGESTIONS

DEBLOCKING TAPES

DITTO does not provide a tape to tape with deblocking function. Blocked card image tapes (i.e., IBM supplied DTR tapes, DOS maintenance tapes, etc.), may be deblocked into an unblocked format, however, by using the tape to card function and assigning SYSPCH to the output tape. The deblocked tape may then be assigned to SYSIPT, eliminating the need in many cases of physically punching the data into cards.

PRINTING TAPE FILES

Programmers who require listings of tape files will find the Tape Hex formats more useful than the character print functions. The Tape Hex functions provide both character and vertical hexidecimal format with a position scale for each tape record. Programmers using these functions under the remote control card operation should also make use of the NBLKS parameter. This optional parameter gives the programmer the capability of obtaining "partial" listings of tape files. This eliminates "lengthy" and "time consuming" tape printing.

PRINTING SYSLST TAPES

When SYSLST is assigned to tape, DOS Job Control statements are written as 121 character records, with the first character being the ASA FORTRAN forms-control character (Type D). DITTO and many other Type III programs which support SYSLST on tape, write 133 character records, with the first character being the CCW operation code for forms-control (Type A). Printing SYSLST tapes presents, therefore, some incompatibilities in forms-control characters to be used.

DITTO provides two (2) functions for printing SYSLST tapes using forms-control characters. TFD assumes all tape records contain the Type D forms-control character. These tape records may contain from 0 to 132 positions of data to be printed.

10. WHAT IS AN INDEXED SEQUENTIAL FILE? Before answering that question, it may be necessary to define a couple of terms. They are physical organization, and logical organization. Physical organization refers to the specific physical layout and placement of records within a file from the standpoint of the hardware. Logical organization, on the other hand, has to do with how the programer or the operating system for that matter, "sees" the records within a file. Physical and logical organization may be exactly the same for any one file, but that is usually not the case when dealing with files which are resident on tape or on a direct access device.

An indexed sequential (ISAM) file is one in which the records are in sequence logically but not necessarily physically. It provides sequential processing capability just as any other sequential file, and in addition, provides the ability to directly access individual records. It must be resident on a direct access storage device (e.g., disk). It differs from a sequential file in that each record within the file contains a specific control field called a key. This key field must be identified to the operating system, and the key within each record must be unique; i.e., no two records in the file can have the same key, just as no two people should have the same social security number.

The total file is composed of at least three separate areas - an index area, a prime data area, and an overflow area. Though these areas are "controlled" by the operating system, the programer can vary the size of each, and, under certain conditions, can look at what is contained in each. Normally, he is not concerned with the details of these areas.

To create or load an indexed sequential file, the records must be in ascending sequence by key. As the records are loaded, the system builds a series of indexes, and places all of the records in the prime data area. Nothing goes into the overflow area during the create process. That area is used only when records are subsequently added to the file, and the system needs space for expansion.

The major advantage to using an indexed sequential file is the great amount of flexibility it provides. When records are added, the whole file does not need to be rewritten as is the case with a straight sequential file. Additionally, as mentioned before, direct accessing and processing of records is possible because of the indexes. Any number of records, in any order, may be read or written without processing the entire file.

Now let us look at the mechanics of creating, updating, and listing indexed sequential files using ANS COBOL. All references are to the sample program attached as Appendix A.

11. REVIEW OF ENVIRONMENT DIVISION AND DATA DIVISION REQUIREMENTS. To process ISAM files, we must describe them properly in the ENVIRONMENT DIVISION and in the DATA DIVISION.

a. ENVIRONMENT DIVISION. SELECT STATEMENT.
(Ref: Lines 54 thru 61.)

(1) The ASSIGN clause.
(Ref: Lines 55 and 58.)

DOS: This clause must be of the form SYSnnn-DA-unit-I in which nnn may be any system number allowed on your system, and unit may be either 2311 or 2314 or 2321. The I identifies the file as being an indexed-sequential file.

OS: This clause must be of the form DA-I-ddname where ddname is the name of the JCL DD card pointing to the indexed-sequential file. This assign identifies the file to the COBOL compiler as being an indexed-sequential file on a direct access device.

(2) The ACCESS clause. When you have an ISAM file, you have the choice of using it as a random file (Direct Access) or as a sequential file. If you wish to use the file as a random file and be able to directly access specific records, you must include the clause ACCESS IS RANDOM in the SELECT statement. If you plan to use the file as a sequential file, you may code either ACCESS IS SEQUENTIAL or omit the ACCESS clause entirely, as SEQUENTIAL is the default. (Ref: Line 59)

(3) The KEY clauses. The RECORD KEY IS clause identifies which field of the ISAM record is the key field. The record key must be specified for any use of an ISAM file. The NOMINAL KEY IS clause is needed whenever ACCESS IS RANDOM is specified. The nominal key is used to identify which record is to be read or written, whereas the record key identifies the one field within the record which is to be used as the key. In other words, when you want a specific record, you must ask for it by its key. This is done by moving the key of the record you want to the nominal key field, and then issuing a READ statement. When accessing an ISAM file sequentially, the nominal key is not required.
(Ref: Lines 56, 60 and 61)

b. DATA DIVISION. FD ENTRIES. The LABEL RECORDS clause in the FD of an ISAM file must be LABEL RECORDS ARE STANDARD. ISAM files may contain only fixed-length records, which may be blocked. As with any other type of file, if the records are blocked, the BLOCK CONTAINS clause must be included as part of the FD.
(Ref: Lines 89 thru 93 and 99 thru 103.)

12. CREATING AN ISAM FILE. (Ref: Lines 208 thru 228) We need two files to create an ISAM file - an input file within which the records, as mentioned before, must be in ascending sequence based on the record key; and an output file which is the ISAM file itself. The Environment Division entry for this file must indicate that access is sequential. The Data Division entry must show LABEL RECORDS ARE STANDARD. In the Procedure Division, the following process is required as a minimum:

a. The ISAM file must be opened as an output file.

b. The records read from the input file must be moved to the record area for the ISAM file. If reformatting is desired, it should be done during this step.

c. To get each record onto the ISAM file, a WRITE statement is needed. The WRITE statement used here is slightly different from that used for straight sequential files. In the event that the records are not actually in sequence, a simple write statement would cause the program to abnormally terminate (ABEND). To prevent this, we can include an INVALID KEY clause with the WRITE statement. The format is as follows:

```
WRITE record-name      FROM data-name

          INVALID KEY Imperative Statement(s)
```

(Ref: Lines 219 thru 226.)

The INVALID KEY is a condition for the WRITE just as AT END is a condition for the READ, and they operate in much the same way. Until the INVALID KEY condition occurs, the statement(s) following the words INVALID KEY are ignored, and control is passed to the next sentence. If the INVALID KEY condition does occur, ONLY then will the statement(s) be executed. The reason for the INVALID KEY option is to prevent us from putting together an invalid file. The INVALID KEY condition will occur whenever we violate one of the three rules concerning building ISAM files.

Rule 1. The records which will make up the ISAM file must be in ascending sequence by their keys. If we attempt to WRITE a record that is out of sequence, an INVALID KEY condition will occur.

Rule 2. The key in each record must be unique. If we attempt to WRITE a record whose key is a duplicate of one we have already written, we also will get an INVALID KEY condition.

Rule 3. Enough disk space must be allocated to contain all records, else an invalid key condition will occur.

The statements following the words INVALID KEY are determined by what you want your program to do when an INVALID KEY condition occurs. You may want to skip over the record that caused the INVALID KEY condition; you may want to print an error message; or you may want to halt the program. Most often, the statements will be a GO TO EXIT or a PERFORM of an error routine.

d. Every record in the file should have a one character field at the very beginning reserved for a deletion code. (Ref: Lines 95 and 105) During creation of the file, the programer is responsible for moving any non-HIGH-VALUE character to this field (Traditionally LOW-VALUE is moved). Any record in the file with HIGH-VALUES in the first character will not be retrieved when processing the file sequentially.

13. UPDATING AN INDEXED SEQUENTIAL FILE. There are two ways to access an existing ISAM file. Let us look at the mechanics of each.

a. RANDOM ACCESS. Accessing a file randomly means that individual records may be selected from the file in any order. The primary benefit in using this method of processing is that not all records have to be read, and processing time can often be reduced substantially. Considerations for random access:

(1) ENVIRONMENT DIVISION. The SELECT statement for the ISAM file must include the ACCESS IS RANDOM, NOMINAL KEY, and RECORD KEY clauses. (Ref: Lines 57 thru 61.)

(2) DATA DIVISION.

(a) In describing the file (FD), the main consideration is to ensure that the block size, record size, and record format are the same as when the file was created.
(Ref: Lines 99 thru 103.)

(b) The data-name associated with the NOMINAL KEY in the SELECT statement must be defined in the WORKING-STORAGE section, preferably as a Level 77 item. It must be the same size, and have the same characteristics as the item within the record which is designated as the RECORD KEY. (Ref: Lines 118, 119.)

(3) PROCEDURE DIVISION. (Ref: LInes 244 thru 287.) Any time an ISAM file is accessed randomly, there must be some means of determining which specific records of the file are to be read. Usually, another input file, called a transaction file, is used to obtain the keys of the records we need. This transaction file generally contains data to update certain records of the ISAM file, and though it may be a physically sequential file, the records

within it do not have to be in any particular order. To handle this type of situation we need to do the following in the Procedure Division:

(a) Open the transaction file as Input.

(b) Open the ISAM file for both Input and Output, because we will read records from that file, update them, and then put them back on the file. Also, we may need to add new records to the file. The OPEN statement looks like this:

OPEN I-O ISAM-file-name

(Ref: Line 176)

(c) A record must now be read from the transaction file. Within that record will be the key of the record to be read from the ISAM file, and it must be moved to the NOMINAL KEY field in working-storage.

(d) The ISAM record may now be retrieved by issuing a READ statement. This particular READ is slightly different from that used with sequential access. Remember, the file is being read randomly, and any record selected may be at the end of the file, in the middle, or at the beginning. In fact, any record may be read more than once in any one processing cycle. So, there is really no end to the number of records read, and an AT END clause is not required with the READ. There is a possibility though, that when we issue a READ for the file, and the system searches for the appropriate record (based on the contents of the NOMINAL KEY), that it may not find that record. If that occurs, an invalid key condition results and unless we provide for handling that condition, the program will ABEND. Fortunately, we can provide for it simply by attaching an INVALID KEY clause to the READ. The format of a READ for random access is:

READ file-name INTO data-name

INVALID KEY statement(s)

(Ref: Lines 250 thru 256.)

The action of the INVALID KEY clause is the same as it was for the WRITE statement; that is, the statement(s) following the words INVALID KEY will not be executed unless an invalid key condition occurs. As stated earlier, that happens when we ask for a particular record, and that record does not exist on the file. It may be that an error of some sort has occurred possibly in source coding or keypunching - and it would then be appropriate to print an error message as well as an image (copy) of the transaction record. Or, if the processing cycle provides for the addition of new records to the file, it may be that this transaction record is one which should be added, and we would want to perform an add-records routine. The procedures to do that will be discussed later.

(e) Assume for now that upon execution of a READ, no invalid key condition was encountered, and the record we asked for has been retrieved from the file, and is now resident in the ISAM record area of main storage. Any required changes may now be made, and when completed, the record is ready to be written back to the file. In all cases so far, when we needed to put a record onto a file, we used a WRITE statement. In this case however, we must use a REWRITE statement. The reason is that the original record is still on the file. (When we read it, a copy of the record was moved to main storage.) In order to put a "new" copy of the record onto the file, we must REWRITE rather than WRITE. The format of the REWRITE is:

```
REWRITE record-name   FROM  data-name

          INVALID KEY statement(s)
```

(Ref: Lines 283 thru 285.)

Notice that we must attach an INVALID KEY clause to the REWRITE. The only reason for an invalid key condition to occur on a rewrite is a change in the contents of the nominal key field between the time the record was read and the time the rewrite was attempted. It is most unlikely that this will happen, but we must provide for it. If it does happen, it is best to print a message indicating what happened (INVALID KEY ON REWRITE), display the record involved, and then terminate the program, as there is evidently a logic error in the program.

(f) In addition to making changes to records, it may be that some records have to be deleted. To indicate which records should be changed, deleted, etc., we can include a type-transaction code in each transaction record. For example, a code of "C" could mean change, a "D" delete, and an "A" add. So far, we have handled only type code "C". The process of deleting records is very similar to that of changing records. The major difference is that we must move HIGH-VALUES to the deletion code field of the ISAM record, which, as pointed out in paragraph 12(d), is a one character field attached to the front of the record at file create time. No other changes need to be made in the record, and it may now be rewritten to the file just as was done with a changed record. Again, (Ref: Lines 270 thru 274) the REWRITE verb must be used instead of WRITE. Though the record is still physically on the file, from a logical standpoint it is not there. It can, however, be retrieved randomly in case it was "deleted" in error.

(g) Now let us look at how we can add new records to an existing ISAM file. An important point we may need to reemphasize here is that the key of each record must be unique. So, if we have to add a new record to the file, it is imperative that there is not already a record on the file with the same key. We can check that in one of two ways. First, let us follow the same procedure as before. We read a record from the transaction file, move the key found there to the NOMINAL KEY area, and then issue a READ to the ISAM file. If an INVALID KEY condition occurs, the record we requested is not on the file. If the type code of the transaction is an "A" then we may add the record to the file using a WRITE statement. (A REWRITE is used only when a record which was previously read has to be put back on the file.) The format of the WRITE is the same as that used in creating the file. Another way to handle the addition of records is to read a record from the transaction file, check the type code to see if it is an "A", and if so, move the transaction record to the ISAM record area, the record key to the nominal key, and then WRITE. If that particular record is not already on the file, the WRITE will be completed successfully (in a very rare case, the file may be full, and an INVALID KEY condition will occur). If there is a record on the file with the same key, an INVALID KEY condition will occur, and it must be handled as an error.

14. SEQUENTIAL RETRIEVAL OF AN INDEXED SEQUENTIAL FILE. (Ref: Lines 305 thru 314) As mentioned before, indexed sequential files can be processed either randomly or sequentially. When updating, it is generally more advantageous to use the random access mode. (To add records to an ISAM file, random access must be used.) Sequential retrieval of records to produce, for example, a printed listing is often necessary. This paragraph will illustrate the process of retrieving sequentially first of all, an entire ISAM file, and secondly, only a portion of that same file. Again let us look at the - considerations by COBOL division.

a. ENVIRONMENT DIVISION. The SELECT statement must include the RECORD KEY IS clause following the ASSIGN TO DA-I-sysname clause. The ACCESS IS SEQUENTIAL clause is optional, but should be included for documentation. If it is desired to start the retrieval process at some point other than the first record, then the NOMINAL KEY IS clause must be included also.

b. DATA DIVISION. The only factors to be considered here are that the blocking factor and record size parameters are specified correctly.

c. PROCEDURE DIVISION.

(1) The file must be opened as an input file only.

(2) If the retrieval is to start at a point other than the beginning of the file (first record), the next step must be to move the RECORD KEY of

the first record to be retrieved to the NOMINAL KEY area, and then issue a START command. The format of the START statement is as follows:

```
START  file-name

          INVALID KEY imperative statement
```

This causes the ISAM file to be searched for the record whose key is contained in the NOMINAL KEY area. If the record is found, a pointer is set so that when a READ statement is given (see next paragraph), the first record read will be the one with the given key, and each subsequent issuance of a READ will result in the next record in sequence being read. If the record requested is not found in the file then the statement following INVALID KEY will be executed. Note that the START statement is not required when retrieval is to begin with the first record of the file.

(3) To actually retrieve a record, a READ statement is needed. The format of this particular READ is the same as that for a straight sequential file, i.e.,

```
READ     file-name

         AT END  imperative statement
```

(Ref: Lines 306 thru 309)

PE-7 EXAMPLE

COBOL (Structured)

I. **INTRODUCTION.** The Supply Officer at Fort Harrison has asked for you to create an ISAM File containing the current Quantity On Hand of the items stocked in the Supply Room. You are to update the ISAM file with each day's Order Records and list the ISAM File after each update.

II. **DEFINITIONS.** ISAM FILE; Indexed Sequential Access Method file.

III. **INPUT/OUTPUT.** INPUT: Input consists of an Order File stored on a 2314 disk pack. The file contains the Create Records for the ISAM File, followed by a blank record, followed by the days Order Records. Assign this file to SYS006. See Record Layouts 1 and 2 for a description of each type of record.

OUTPUT: Output will consist of two (2) listings.

1. A Transaction Listing showing each transaction as it is processed against the ISAM File with one of the following messages;
   - A. TRANSACTION FOR DELETED RECORD-NOT PROCESSED
   - B. UNMATCHED PART NUMBER
   - C. DELETED RECORD
   - D. NEGATIVE BALANCE ON HAND
   - E. PROCESSED

   See Form Layout #1 for a description of this report.

2. An ISAM FILE LISTING to list all records on the ISAM File by part number showing the updated Quantity On Hand. See Form Layout #2 for a description of this report.

Assign these reports to SYS005.

IV. **ASSUMPTIONS.** There will be more than one (1) page of output, so produce a heading at the top of each page.
CUSTOMER CODE will also contain a 'D' if the record is to be deleted from the ISAM File.
During the updating of the ISAM File no records are to be added.

V. METHODS.

1. Read the CREATE RECORDS and create the ISAM File. If you run out of space on the disk, or the records are out of sequence, stop the processing at that point

2. Read the ORDER RECORDS, find the matching ISAM record, and subtract the Quantity Ordered from the Quantity On Hand.
If the record has already been deleted, annotate it with MESSAGE 'A'.
If a matching record is not found, annotate it with MESSAGE 'B'.
If the CUSTOMER CODEis equal to a 'D', delete the record andannotate it with MESSAGE 'C'.
If the updated Quantity On Hand happens to be NEGATIVE, annotate it with MESSAGE 'D'.
Annotate all other transactions with MESSAGE 'E'.

3. List the ISAM File after it is updated.

Both reports will consist of twenty (20) double spaced lines per page.

VI. EDIT FACTORS.

All numeric fields will contain valid numeric data.

IA-03-05-14 (D) "For Instructional Purposes Only" S255

RECORD LAYOUT #2 1

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO 1 OF 1 PAGES

DATE December 19xx

SYSTEM ID Truck Supply System

RECORD LENGTH 80 Characters

PREPARED BY SP6 D.W. Gross

☐ CARD ☒ DISK ☐ TAPE ☐ OTHER

FILE ID SYS006

1 record per block

REMARKS ORDERS FILE (Create Records)

RELATIVE POSITION	IDENTIFICATION OF ELEMENT *(FIELD)*	ABBREVIATION	LENGTH/ CLASS	LOCATION
1	PART NUMBER		4N	1-4
2	Quantity On Hand		4N	5-8
3	Not Used		72AN	9-80

400

IA-03-05-14 (D) "For Instructional Purposes Only" S255

RECORD LAYOUT *For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.*	PAGE NO 1 OF 1 PAGES DATE December 19xx
SYSTEM ID: Truck Supply System	RECORD LENGTH: 80 Characters
PREPARED BY: SP6 D.W. Gross	FILE ID: SYS006
☐ CARD ☒ DISK ☐ TAPE ☐ OTHER	
1 record per block	REMARKS: ORDERS FILE (Order Record)

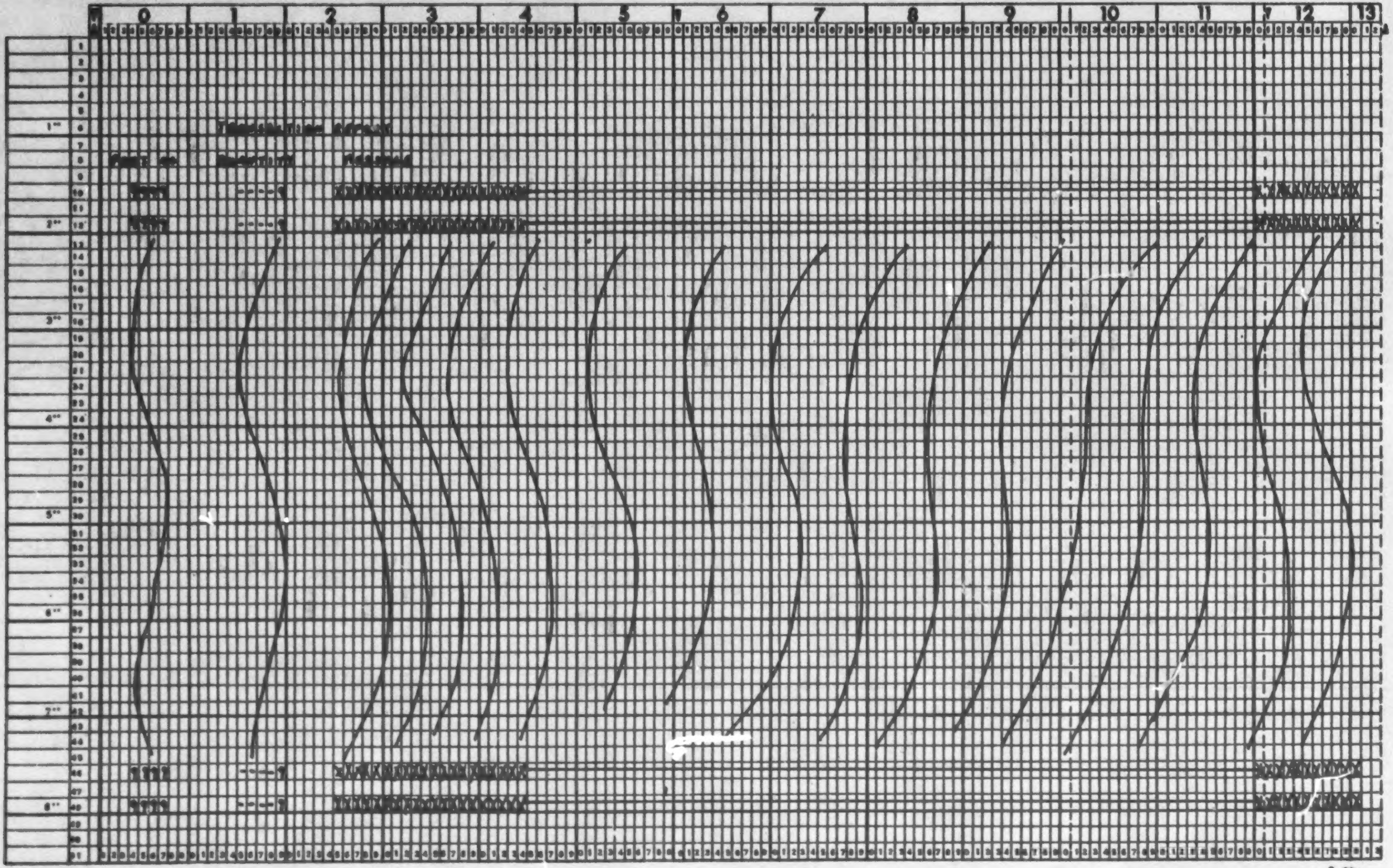
RELATIVE POSITION	IDENTIFICATION OF ELEMENT *(FIELD)*	ABBREVIATION	LENGTH/ CLASS	LOCATION
1	PART NUMBER			
1a	CLASS NUMBER (either a 1,2, or 3)		1N	1
1b	Part Number		3N	2-4
2	CUSTOMER NAME		18AN	5-22
3	CUSTOMER ADDRESS		18AN	23-40
4	UNIT OF ISSUE		2AN	41-42
5	QUANTITY ORDERED		3N	43-45
6	PRICE PER ITEM		5N*	46-50
7	CUSTOMER CODE		1AN**	51 ***
8	NOT USED		29AN	52-80

* DOLLARS AND CENTS FIELD

** CAN ONLY BE A BLANK OR A '?'

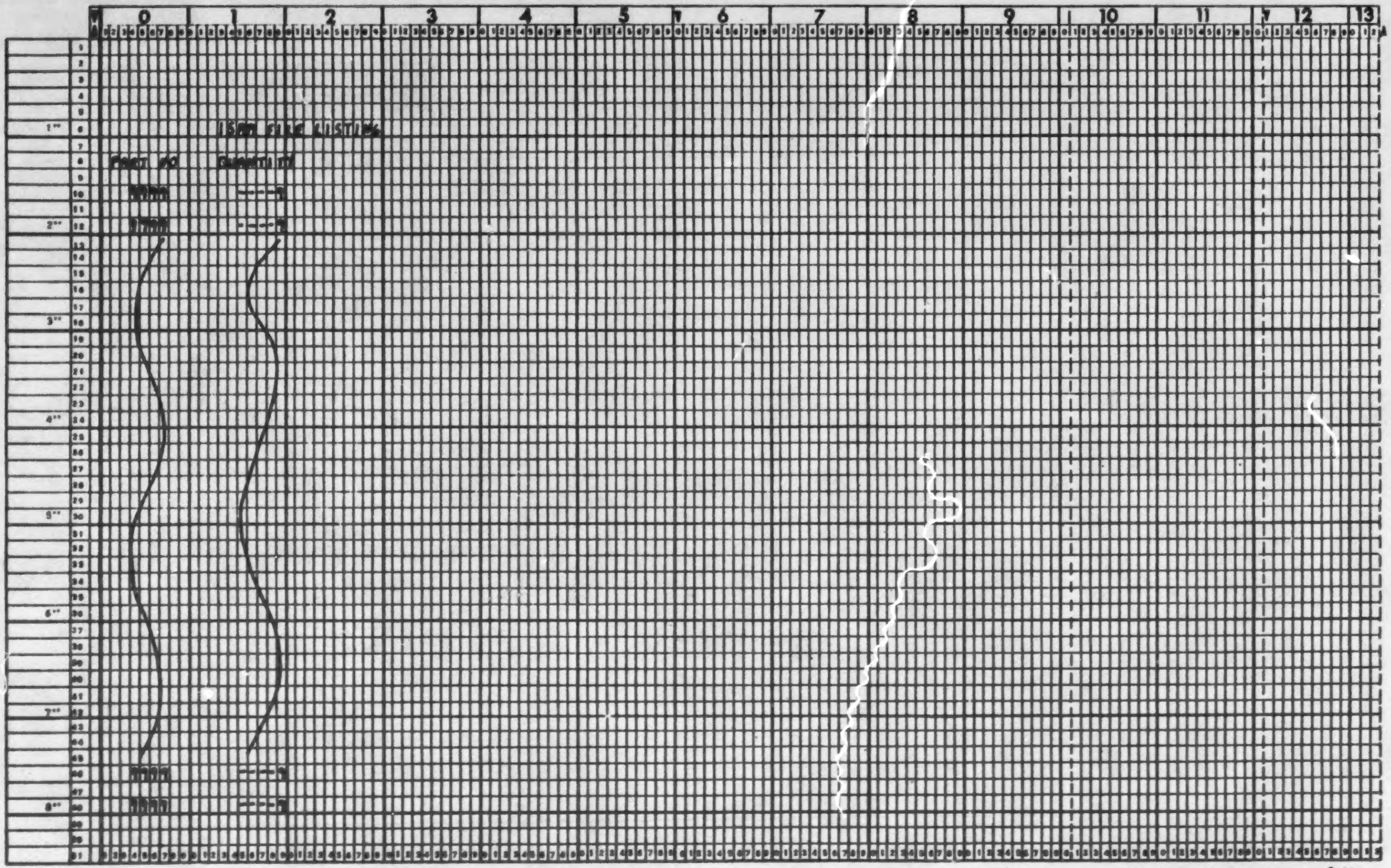
*** Will be a 'D' for deletion transactions

FORM LAYOUT #1


0
1
2
3
4
5
6
7
8
9
10
11
12
13
1"
2"
3"
4"
5"
6"
7"
8"

104

FORM LAYOUT #2



1500 FILE LISTING

PART NO QUANTITY

99999 ZZZZ9

99999 ZZZZ9

99999 ZZZZ9

99999 ZZZZ9

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

6

```
                 IDENTIFICATION DIVISION.
                 PROGRAM-ID.     PE-7EXAMPLE.
                 AUTHOR.         SP6DW GROSS
                 INSTALLATION.   CSD, USAIA, FT HARRISON IN 46216.
                 DATE-WRITTEN.   MAR 1, 1978
                 DATE-COMPILED.  JUL 5, 1978
                 SECURITY.       NONE
                 REMARKS.  ***********************************************
                *                   GENERAL FUNCTIONS.                          *
                *                       THIS PROGRAM ILLUSTRATES                *
                *                   1   CREATING AN ISAM FILE                   *
                *                   2   RANDOMLY UPDATING AN ISAM FILE          *
                *                   3   READING AND LISTING THE UPDATED ISAM    *
                *                       FILE SEQUENTIALLY.                      *
                *                                                               *
                *                   SPECIFIC FUNCTIONS.                         *
                *                                                               *
                *                       THIS PROGRAM READS THE CREATE RECORDS   *
                *                   AND BUILDS AN ISAM FILE. THE ORDER RECORDS  *
                *                   FOR EACH DAY ARE THEN MATCHED AGAINST THE   *
                *                   BALANCE FILE (ISAM FILE), AND THE QUANTITY  *
                *                   ORDERED IS SUBTRACTED FROM THE BALANCE.     *
                *                   FILES AUANTITY ON HAND. IF A NEGATIVE RE-   *
                *                   SULT IS REACHED, A MESSAGE IS PRINTED TO    *
                *                   THAT EFFECT ON THE TRANSACTION REPORT.      *
                *                       FINALLY THIS PROGRAM LISTS THE UPDATED  *
                *                   BALANCE FILE.                               *
                *****************************************************************
```

```
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-360-H40.
  OBJECT-COMPUTER. IBM-360-H40.
  SPECIAL-NAMES.
      COI IS TOP-OF-PAGE.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
      SELECT IM-ORDER-FILE
          ASSIGN TO UT-S-SYS006.
      SELECT OR-PRINTED-REPORTS
          ASSIGN TO UT-S-SYS005.
********************************************************************
*                                                                  *
*                      NOTICE THE ASSIGN CLAUSE FOR THE FOL-       *
*                 LOWING SELECTS.  ISAM FILES MUST BE ASSIGN-      *
*                 ED TO DIRECT ACCESS DEVICES, THUS THE 'DA        *
*                 -I'.  ALSO YOU MUST SPECIFY WHICH FIELD IN       *
*                 THE RECORDS IS THE KEY FIELD.  FOR RANDOM        *
*                 PROCESSING, THE CLAUSES ACCESS IS RANDOM,        *
*                 AND NOMINAL KEY IS ..... MUST BE SPECIFIED.      *
*                                                                  *
********************************************************************
     SELECT BM-BALANCE-FILE
         ASSIGN TO DA-I-SYSC08
         RECORD KEY IS OM01-PART-NUMBER.
     SELECT 8M-BALANCE-FILE-RANDOM
         ASSIGN TO DA-I-SYS008
         ACCESS IS RANDOM
         RECORD KEY IS 8M02-PART-NUMBER
         NOMINAL KEY IS WS-PART-NUMBER
```

```
DATA DIVISION.
FILE SECTION.
FD  IM-ORDER-FILE
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORDS ARE IM01-CREATE-REC
                     IM02-ORDER-REC.
01  IM01-CREATE-REC.                                        PIC 9(4).
    05  IM01-PART-NUMBER                                    PIC S9(4).
    05  IMOL-QUANTITY                                       PIC X(72).
    05  FILLER
01  IM02-ORDER-REC.                                         PIC 9(4).
    05  IM02-PART-NUMBER                                    PIC X(18).
    05  IM02-CUSTOMER-NAME                                  PIC X(18).
    05  IM02-CUSTOMER-ADDRESS                               PIC XX.
    05  IM02-UNIT-OF-ISSUE                                  PIC 999.
    05  IM02-QUANTITY-ORDERED                               PIC 999V99.
    05  IM02-PURCHASE-AMOUNT                                PIC X.
    05  IM02-CUSTOMER-CODE                                  PIC X(29).
    05  FILLER
FD  OR-PRINTED-REPORTS
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 133 CHARACTERS
    LABEL RECORD IS STANDARD                                PIC X(133).
    DATA RECORD IS OR01-PRINT-LINE.
01  OR01-PRINT-LINE
FD  BM-BALANCE-FILE
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 11 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORD IS BM01-BLAANCE-REC.                        PIC X.
01  BM01-BALANCE-REC.                                       PIC 9(4).
    05  BM01-DELETE-KEY                                     PIC XX.
    05  BM01-PART-NUMBER                                    PIC S9(4).
    05  FILLER
    05  BM01-QUANTITY-ON-HAND
FD  BM-BALANCE-FILE-RANDOM
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 11 CHARACTERS
    LABEL RECORD IS STANDARD                                PIC X.
    DATA RECORD IS BM02-BALANCE-REC-R.                      PIC 9(4).
01  BM02-BALANCE-REC-R.                                     PIC XX.
    05  BM02-DELETE-KEY                                     PIC S9(4).
    05  BM02-PART-NUMBER
    05  FILLER
    05  BM02-QUANTITY-ON-HAND
```

```
       WORKING-STORAGE SECTION.
       77  WS-BEGIN-POINT                                   PIC X(27)
                                VALUE 'WORKING STORAGE BEGINS HERE'.
       77  WS-EOF-SWITCH                                    PIC XXX
                                VALUE 'OFF'.
       77  WS-ERROR-SW                                      PIC XXX
                                VALUE 'OFF'.
       77  WS-LINE-COUNTER                                  PIC 99
                                VALUE 26.
       77  WS-PART-NUMBER                                   PIC 9(4)
                                VALUE 0.
       01  WS-HEADING-LINE-1.
           05  FILLER                                       PIC X(13)
                                VALUE SPACES.
           05  WS-HL1-TITLE                                 PIC X(18).
           05  FILLER                                       PIC X(102)
                                VALUE SPACES.
       01  WS-HEACING-LINE-2.
           05  FILLER                                       PIC XX
                                VALUE SPACES.
           05  FILLER                                       PIC X(7)
                                VALUE 'PART NO'.
           05  FILLER                                       PIC X(4)
                                VALUE SPACES.
           05  FILLER                                       PIC X(8)
                                VALUE 'QUANTITY'.
           05  FILLER                                       PIC X(5)
                                VALUE SPACES.
           05  WS-HL2-MESSAGE                               PIC X(7)
                                VALUE 'MESSAGE'.
           05  FILLER                                       PIC X(100)
                                VALUE SPACES.
       01  WS-DETAIL-LINE.
           05  FILLER                                       PIC X(4)
                                VALUE SPACES.
           05  WS-DL-PART-NUMBER                            PIC 9(4).
           05  FILLER                                       PIC X(7)
                                VALUE SPACES.
           05  WS-DL-QUANTITY                               PIC ----9.
           05  FILLER                                       PIC X(5)
                                VALUE SPACES.
           05  WS-DL-MESSAGE                                PIC X(106).
       01  WS-END-POINT                                     PIC X(25)
                                VALUE 'WORKING STORAGE ENDS HERE'.
```

```
      PROCEDURE DIVISION.
      0000-DRIVER.
     ***************************************************************
     *                                                             *
     *                   THE DRIVER PORTION OF THIS PROGRAM        *
     *               DIRECTS THE EXECUTION THROUGH THE THREE       *
     *               PARTS OF THIS PROGRAM.                        *
     *                                                             *
     ***************************************************************
           OPEN INPUT IM-ORDER-FILE
               OUTPUT BM-BALANCE-FILE.
           PERFORM 0100-CREATE-ROUTINE THRU 0100-EXIT
               UNTIL WS-EOF-SWITCH EQUAL 'ON'.
           IF WS-ERROR-SW EQUAL 'ON'
               GO TO 0000-EXIT.
           CLOSE BM-BALANCE-FILE.
     ***************************************************************
     *                                                             *
     *                   THE FOLLOWING OPEN STATEMENT ILLUSTRA-    *
     *               TES THE 'I-O' OPTION. THIS MEANS THAT THE     *
     *               FILE WILL BE OPENED AS BOTH INPUT AND OUTPUT.*
     *                                                             *
     ***************************************************************
           OPEN I-O BM-BALANCE-FILE-RANDOM
               OUTPUT OR-PRINTED-REPORTS.
           MOVE 'OFF' TO WS-EOF-SWITCH.
           MOVE 'TRANSACTION REPORT' TO WS-HL1-TITLE.
           PERFORM 0200-UPDATE-ROUTINE THRU 0200-EXIT
               UNTIL WS-EOF-SWITCH EQUAL 'ON'.
           CLOSE IM-ORDER-FILE
               BM-BALANCE-FILE-RANDOM.
           OPEN INPUT BM-BALANCE-FILE
           MOVE 'OFF' TO WS-EOF-SWITCH.
           MOVE 26 TO WS-LINE-COUNTER.
           MOVE SPACES TO WS-DL-MESSAGE.
           MOVE SPACES TO WS-HL2-MESSAGE.
           MOVE 'ISAM FILE LISTING' TO WS-HL1-TITLE.
           PERFORM 0500-LIST-ROUTINE THRU 0500-EXIT
               UNTIL WS-EOF-SWITCH EQUAL 'ON'.
           CLOSE OR-PRINTED-REPORTS
           BM-BALANCE-FILE.
       0000-EXIT.
           STOP RUN.
      ****************************************************************
      *                                                              *
      *             THE CREATE ROUTINE READS EACH INPUT              *
      *         CREATE RECORD, MOVES THE FIELDS TO THE               *
      *         ISAM BALANCE RECORD, AND WRITES EACH REC-            *
      *         ORD. NOTICE THE WRITE STATEMENT. WHEN                *
      *         WRITING TO AN ISAM FILE THE INVALID KEY              *
      *         CLAUSE IS USED. IN THIS CASE IF THERE IS             *
      *         AN INVALID KEY IT MEANS THERE IS NO MORE             *
      *         SPACE LEFT ON THE DISK.                              *
      *                                                              *
      ****************************************************************
       0100-CREATE-ROUTINE.
           READ IM-ORDER-FILE
               AT END
                   MOVE 'ON' TO WS-EOF-SWITCH
                   GO TO 0100-EXIT.
           IF IM01-CREATE-REC EQUAL SPACES
               THEN MOVE 'ON' TO WS-EOF-SWITCH
                   GO TO 0100-EXIT.
           MOVE LOW-VALUES TO BM01-DELETE-KEY.
           MOVE IM01-PART-NUMBER TO BM01-PART-NUMBER.
           MOVE IM01-QUANTITY TO BM01-QUANTITY-ON-HAND.
           WRITE BM01-BALANCE-REC
               INVALID KEY
                   DISPLAY 'FULL DISK OR TRANS OUT OF SEQ',
                       IM01-PART-NUMBER
                   CLOSE IM-ORDER-FILE
                       BM-BALANCE-FILE
                   MOVE 'ON' TO WS-EOF-SWITCH
                   MOVE 'ON' TO WS-ERROR-SW.
       0100-EXIT.
           EXIT.
      ****************************************************************
      *                                                              *
      *             THE FUNCTION OF THE UPDATE ROUTINE IS            *
      *         TO READ EACH ORDER RECORD, READ THE ISAM             *
      *         FILE RANDOMLY FINDING THE MATCHING RECORD            *
      *         AND ADJUST THE QUANTITY ON HAND. THEN IN-            *
      *         VALID KEY CLAUSE IS USED WITH THE READ               *
      *         STATEMENT, WHEN THE MATCHING RECORD IS NOT           *
      *         THERE, THIS CONDITION IS INVOKED.                    *
      *             AFTER THE QUANTITY ON HAND IS ADJUSTED,          *
      *         THE ISAM RECORD IS REPLACED IN ITS PLACE             *
      *         ON THE FILE. THE REWRITE STATEMENT IS USED           *
      *         FOR THIS.                                            *
      *                                                              *
      ****************************************************************
       0200-UPDATE-ROUTINE.
           READ IM-ORDER-FILE
               AT END
                   MOVE 'ON' TO WS-EOF-SWITCH
                   GO TO 0200-EXIT.
           MOVE IM02-PART-NUMBER TO WS-PART-NUMBER.
           READ BM-BALANCE-FILE-RANDOM
               INVALID KEY
                   MOVE IM02-PART-NUMBER TO WS-DL-PART-NUMBER
               MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
               MOVE 'UNMATCHED PART NUMBER' TO WS-DL-MESSAGE
               PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
               GO TO 0200-EXIT.
     IF BM02-DELETE-KEY EQUAL HIGH-VALUES
          THEN MOVE 'TRANSACTION FOR DELETED RECORD-NOT PROCESSED'
               TO WS-DL-MESSAGE
               MOVE IM02-PART-NUMBER TO WS-DL-PART-NUMBER
               MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
               PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
               GO TO 0200-EXIT.
     IF IM02-CUSTOMER-CODE EQUAL 'D'
          THEN MOVE HIGH-VALUES TO BM02-DELETE-KEY
               MOVE 'DELETED RECORD' TO WS-DL-MESSAGE
               MOVE IM02-PART-NUMBER TO WS-DL-PART-NUMBER
               MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
               PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
               REWRITE BM02-BALANCE-REC-R
                   INVALID KEY
                        DISPLAY 'RECORD NOT DELETED',
                             IM02-PART-NUMBER
               GO TO 0200-EXIT.
     SUBTRACT IM02-QUANTITY-ORDERED FROM BM02-QUANTITY-ON-HAND.
     MOVE BM02-PART-NUMBER TO WS-DL-PART-NUMBER.
     MOVE BM02-QUANTITY-ON-HAND TO WS-DL-QUANTITY.
     IF BM02-QUANTITY-ON-HAND IS LESS THAN 0
          THEN MOVE 'NEGATIVE BALANCE ON HAND' TO WS-DL-MESSAGE
          ELSE
               MOVE 'PROCESSED' TO WS-DL-MESSAGE.
     PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT.
     REWRITE BM02-BALANCE-REC-R
          INVALID KEY
               DISPLAY BM02-PART-NUMBER, WS-PART-NUMBER.
0200-EXIT.
     EXIT.
0300-PRINT-ROUTINE.
     IF WS-LINE-COUNTER GREATER THAN 19
          THEN PERFORM 0400-HEADER-ROUTINE THRU 0400-EXIT.
     WRITE OR01-PRINT-LINE FROM WS-DETAIL-LINE
          AFTER ADVANCING 2 LINES.
     ADD 1 TO WS-LINE-COUNTER.
     MOVE SPACES TO WS-DETAIL-LINE.
0300-EXIT.
     EXIT.
0400-HEADER-ROUTINE.
     MOVE ZERO TO WS-LINE-COUNTER.
     WRITE OR01-PRINT-LINE FROM WS-HEADING-LINE-1
          AFTER ADVANCING TOP-OF-PAGE.
     WRITE OR01-PRINT-LINE FROM WS-HEADING-LINE-2
          AFTER ADVANCING 2 LINES.
0400-EXIT.
     EXIT.
0500-LIST-ROUTINE.
     READ BM-BALANCE-FILE
          AT END
               MOVE 'ON' TO WS-EOF-SWITCH
               GO TO 0500-EXIT.
```

```
00310             MOVE BM01-QUANTITY-ON-HAND TO WS-DL-QUANTITY.
00311             MOVE BM01-PART-NUMBER TO WS-DL-PART-NUMBER.
00312             PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT.
00313         0500-EXIT.
00314             EXIT.
```

```
              TRANSACTION REPORT
PART NO      QUANTITY      MESSAGE
  1941          -35       NEGATIVE BALANCE ON HAND
  1268          993       PROCESSED
  1767         2513       PROCESSED
  1513         6749       PROCESSED
  1052         5048       PROCESSED
  1223           -9       NEGATIVE BALANCE ON HAND
  1454         4256       PROCESSED
  1689         9903       PROCESSED
  1169          394       PROCESSED
  1491          154       PROCESSED
  1767         2507       PROCESSED
  1268          981       PROCESSED
  1513         6740       PROCESSED
  1941         -285       NEGATIVE BALANCE ON HAND
  1655          117       PROCESSED
  1689         9901       PROCESSED
  1655           53       PROCESSED
  1454         4239       PROCESSED
  1052         5047       PROCESSED
  1513         6738       PROCESSED
```

```
                TRANSACTION REPORT
PART NO         QUANTITY        MESSAGE
  1689            9900        PROCESSED
  1491             129        PROCESSED
  1767            2499        PROCESSED
  1454            4237        PROCESSED
  1169             384        PROCESSED
  1223             -29        NEGATIVE BALANCE ON HAND
  2086            9742        PROCESSED
  2019            9299        PROCESSED
  2652             426        PROCESSED
  2152            7341        PROCESSED
  2925            1415        PROCESSED
  2068             927        PROCESSED
  2529            5827        PROCESSED
  2019            9298        PROCESSED
  2652             414        PROCESSED
  2152            7332        PROCESSED
  3109             999        PROCESSED
  3327            1501        PROCESSED
  3007            3579        PROCESSED
  3318            7800        PROCESSED
```

```
                   TRANSACTION REPORT
PART NO       QUANTITY       MESSAGE
  3765          4987       PROCESSED
  3082          9921       PROCESSED
  3007             3       DELETED RECORD
  3007          3576       PROCESSED
  3007             2       TRANSACTION FOR DELETED RECORD-NOT PROCESSED
  3318          7799       PROCESSED
  3327          1498       PROCESSED
  3765          4986       PROCESSED
  3109           959       PROCESSED
  3082          9920       PROCESSED
  3327          1490       PROCESSED
  3765          4985       PROCESSED
  9999             3       UNMATCHED PART NUMBER
  3109           938       PROCESSED
```

```
          ISAM FILE LISTING

PART NO   QUANTITY

  1052      5047

  1169       384

  1223       -29

  1268       981

  1454      4237

  1491       129

  1513      6738

  1655        53

  1689      9900

  1767      2499

  1941      -285

  2019      9298

  2068       927

  2086      9742

  2152      7332

  2529      5827

  2652       414

  2925      1415

  3082      9920

  3109       938
```

```
          ISAM FILE LISTING

PART NO       QUANTITY

  3318          7799

  3327          1490

  3765          4985
```

416

IA-01-02-10 (D)
IA-01-03-10 (D)

Assembler Language

Practical Exercises

ALC Team, Software Division
Data Processing Department
United States Army Institute
of Administration

417

DC's and DS's

ALC PE #1

A. Write the statements necessary to define the heading shown on the attached form layout. (Form Layout 1)

B. Write the statements necessary to define the detail line shown on the attached form layout. (Form Layout 1)

C. Write the statements necessary to define an input area corresponding to the format shown on the attached card layout. (Card Layout 1)

D. Write the statement to allocate enough space for the printer output area.

INSTRUCTIONAL USE ONLY

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

SYSTEM
CARD LAYOUT 1

PREPARED BY

NO. OF PAGES

		SSN	NAME		RANK		DATE OF RANK	NOT USED
	1 2 3	4 5 6 7 8 9 10 11 12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	31 32	33 34 35	36 37 38 39 40 41	42 43 44 45 46 47	48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

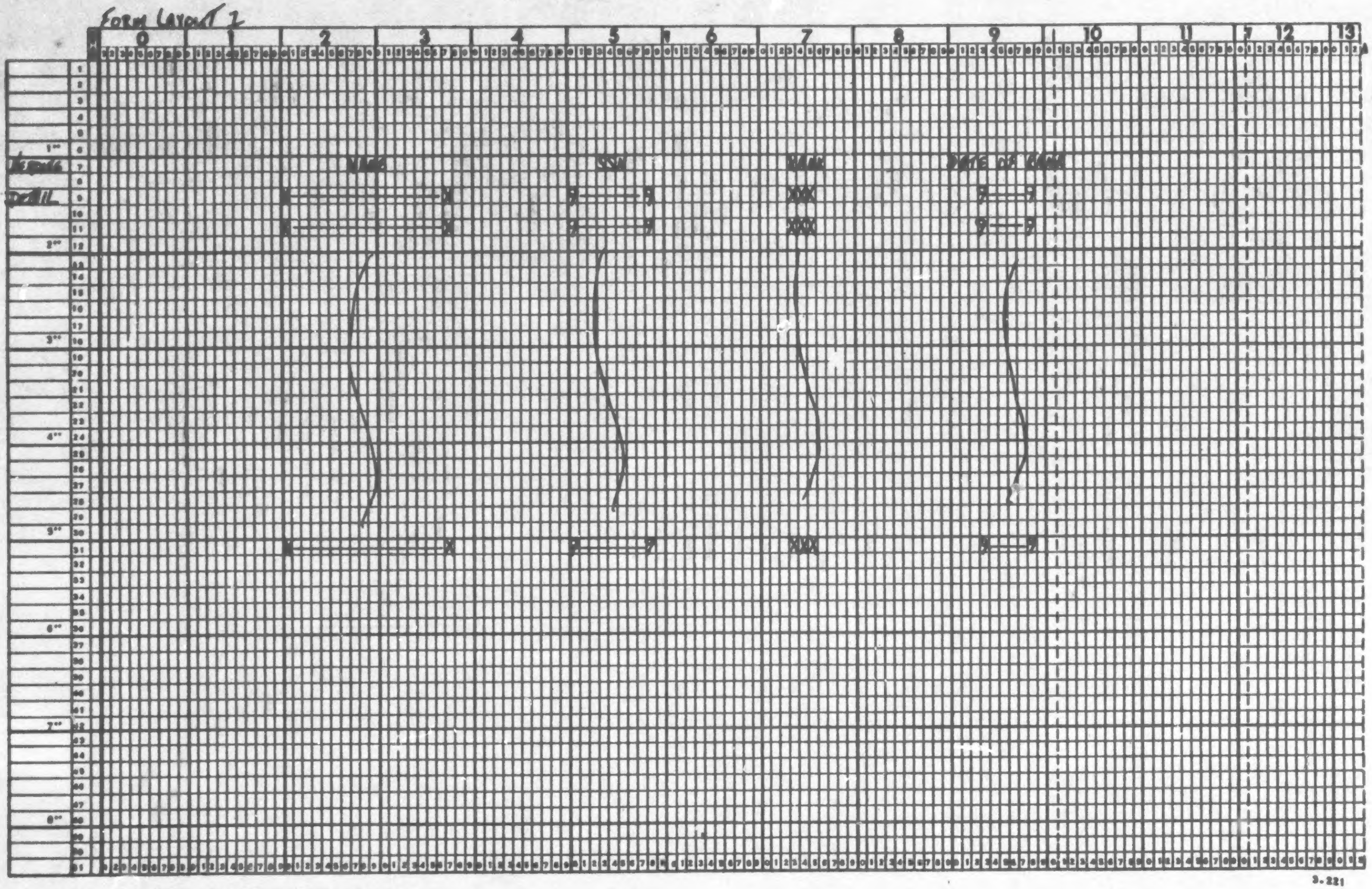
DA FORM 1 AUG 66 3165

* U.S. GOVERNMENT PRINTING OFFICE: 1970 — 381-007

418

FORM LAYOUT

FORM LAYOUT 2


0
1
2
3
4
5
6
7
8
9
10
11
12
13
1"
HEADING
DETAIL
2"
3"
4"
5"
6"
7"
8"
NAME
SSN
RANK
DATE OF RANK
X——X
9——9
XXX
9——9

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

419

SNOOPY **EXECUTE All**
80-132
ALC PE # 2

One of the programmers in your shop has left you with a partially written ALC program. He has completed the logic portion of the program and all that remains to be written is the declarative section and the DTF's.

You are to define the following areas in storage:

1. A one byte field named SPACE initialized to blank.
2. A 132 byte field named OUTPUT, no initial value is needed. This field will be used to hold character information.
3. An 80 byte field called INPUT. This field should be defined with a 0 duplication factor.
4. A 79 byte field with no name attached. This field will hold character information, and no initial value is needed.
5. A one byte field named CODE. It will hold character data and needs no initial value.
6. A one byte field called DASH initialized to the value '-'.

The following DTF's are also needed:

1. A DTFPR. The I/O area is OUTPUT, the device address is SYSLST, the file name is PRINT, and the CNTRL macro will be used, the block size should be 132.

2. A DTFCD. The I/O area is INPUT, the device address is SYSIPT, the file name is CARD, and the end of file address is ENDJOB.

The above storage areas and DTF's should be coded in the order shown above. Once you have done the required coding, keypunch the cards and submit them for execution. The procedural part of the program will be supplied by the student job stream as will the data cards.

PERSONNEL LISTING

ALC PE #3

Assemble: A01
Execute: A03

You are required to write an ALC program to list all of the people in your unit.

INPUT: The input has the same format as the input for ALC PE #1. The logical name for the card reader is SYSIPT.

OUTPUT: The heading and detail lines will have the same format as those for ALC PE #1. You may assume that all output will fit on one page. The logical name for the printer is SYSLST. Double space all output.

ASSEMBLE A01
EXECUTE A03

ALC PE 4

OFFICE LISTING

I. Introduction: You are to write a program that will print a listing of the people assigned to the office. This listing will show each individual's name, phone number, home address, and marital status.

II. Definitions: SCODE = The record selection code
MCODE = The marital status code

III. Input/Output:

A. Input: The input will be the personnel master file which is on cards (see attached card layout). **Device address = SYSIPT.**

B. Output: The output will be a printed report on the 1403 line printer (see the attached form layout). **Device address = SYSLST.**

IV. Assumptions:

A. The listing will fit on one page

B. The MCODE will be only an M, S, or D

V. Methods:

A. Provide a title and column headings as shown on the form layout

B. List only those personnel that have an SCODE of zero. Use the CLC instruction to check the SCODE

C. Using an MVI instruction, insert a hyphen between the third and fourth digit of the phone number on the printed report. This will require breaking the phone number down into two parts.

D. Determine the individuals marital status using the CLI instruction. If the code is an "M",print "MARRIED" in the STATUS field; if the code is an "S", print "SINGLE"; and if the code is a "D", print "DIVORCED".

VI. Edit Factors: None

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO. 1

NO. OF PAGES 1

SYSTEM

Card Layout 2

PREPARED BY

	Phone Number	Name		S CODE		M CODE	Address
1 2 3 4 5	6 7 8 9 10 11 12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37	38	39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59	60	61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

8

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

423

DA FORM 1 AUG 70 3165

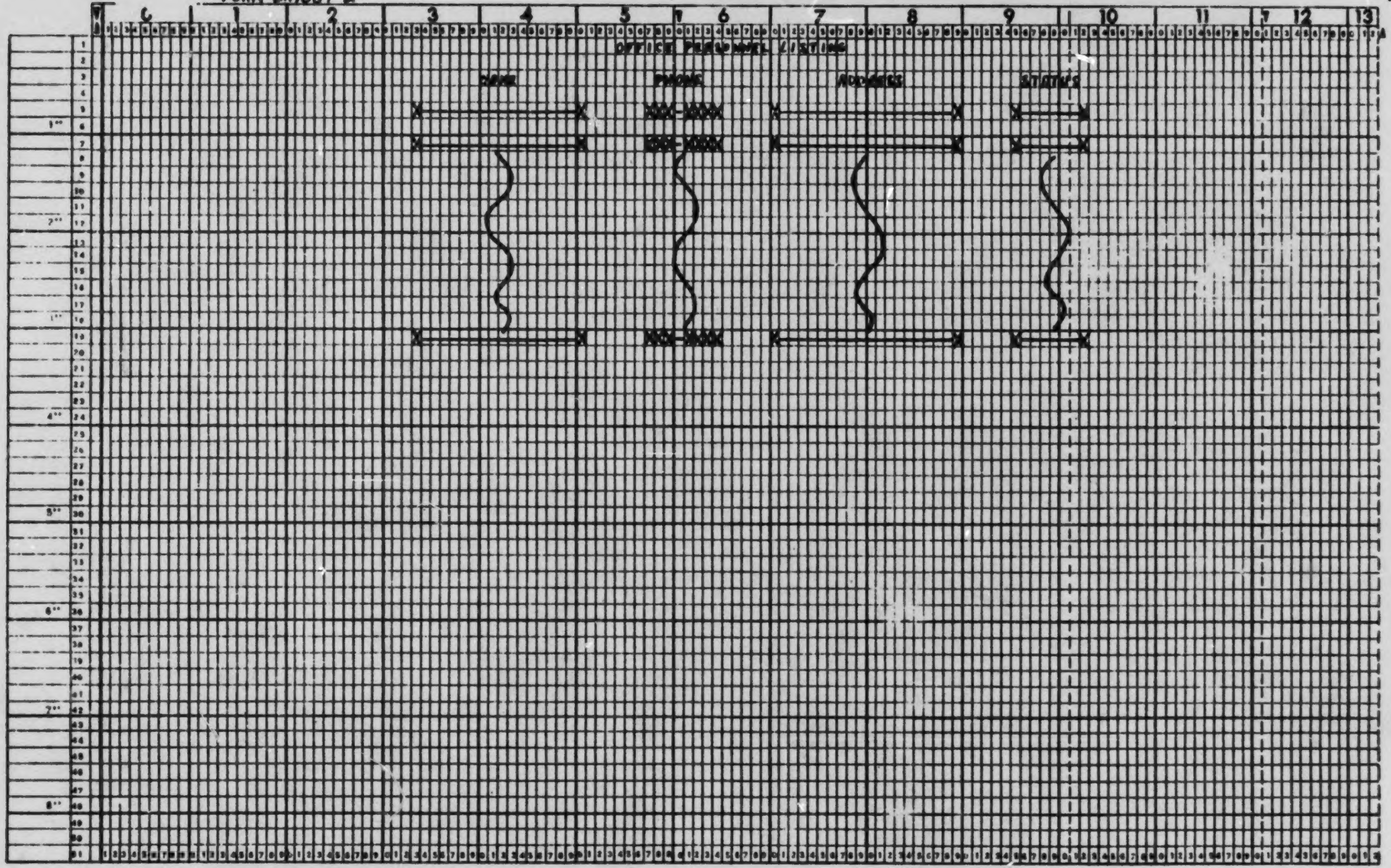
U.S. GOVERNMENT PRINTING OFFICE: 1970 — 301-687

477

424

FORM LAYOUT

FORM LAYOUT 2


OFFICE PERSONNEL LISTING
NAME
PHONE
ADDRESS
STATUS
XXX-XXXX
XXX-XXXX
XXX-XXXX

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

OVER 36

ALC PE 5

Assemble: A01
Execute: A03

You are to write an ALC program to produce a listing of all personnel in the battalion who have completed more than 35 months of service.

INPUT: The input is a personnel master file presently stored on punched cards. See Card Layout 3.

OUTPUT:
1) The output is on a 132 print position printer.
2) Print a heading at the top of each new page.
3) Only 10 detail lines should be printed per page.
4) Print a total line containing the total personnel with 36 or more months of service.
5) Double space all lines except triple space between the total line and the last line.
6) See Form Layout 3 for various line formats.

Assumptions: Not more than 999 persons will have more than 35 months of service. Since we can record only full months of service, the person must have 36 months of service or more to qualify.

Methods:
a) Read each card and print a detail line only for personnel with over 35 months of service.
b) Keep a count of the total number of personnel with more than 35 months of service. This total will be part of the final total line.
c) Do not edit the months of service figure. However, the total personnel figure should be edited with leading nonsignificant zeros suppressed except when the figure is 000 in which case the total figure should print as a single zero.

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

INSTRUCTIONAL USE ONLY

PAGE NO.

SYSTEM

CARD LAYOUT 2

PREPARED BY

NO. OF PAGES

Social Security Number

Name

ETS

Date of Rank

Not Used

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Company

Rank

Number of Dependents

Battalion

Pay Grade

Months of Service

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

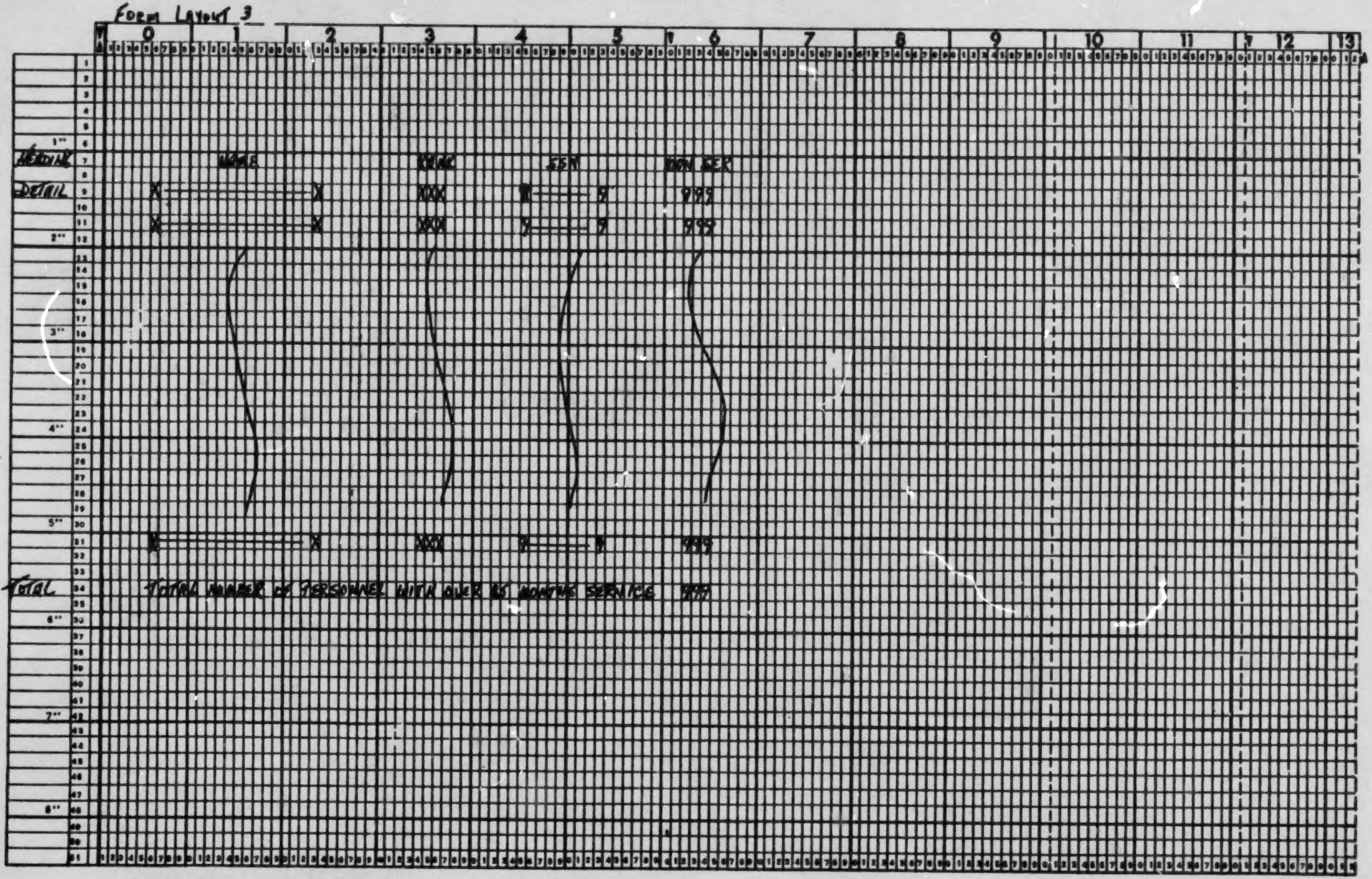
DA FORM 1 AUG 64 3165

U.S. GOVERNMENT PRINTING OFFICE: 1970 - 381-047

426

FORM LAYOUT

FORM LAYOUT 3



HEADING
NAME
[illegible]
SSN
MON SER
DETAIL
X——X XXX 9——9 999
X——X XXX 9——9 999
X——X XXX 9——9 999
TOTAL
TOTAL NUMBER OF PERSONNEL WITH OVER [illegible] MONTHS SERVICE 999

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

ASSEMBLE A01
EXECUTE A04

ALC PE 6

PAYROLL EXERCISE

I. Introduction: You are to write a program that will calculate the net pay and withholding tax for each individual within the company. The report that the program generates will list each individual and the information pertaining to his pay. It will also show the total tax withheld for each department.

II. Definitions: None

III. Input/Output:

A. Input: The input is on cards which are sorted by department. The pay and tax rate fields have two implied decimal points (see attached card layout for input data format). Device address = SYSIPT.

B. Output: The output will be on a 1403 line printer and will consist of column headings on each page, detail lines, and a total line for each department. Start each department on a new page (see attached form layout for output specifications). Device address = SYSLST.

IV. Assumptions:

Each department will fit on one page. The total tax figure will not exceed $99999.99.

V. Methods:

A. For each record calculate the withholding tax as follows:
Withholding tax = Tax Rate x Gross Pay

B. Calculate Net Pay as follows:
Net Pay = Gross Pay - Withholding Tax

C. Keep track of the total taxes for each department. This will be part of the total line.

D. Zero suppress all numeric output fields. The Gross Pay, Withholding Tax, and Net Pay fields should have decimal points, commas, and fixed dollar signs as shown on the form layout.

VI. Edit factors: None

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO. 1

SYSTEM Card Layout 5

PREPARED BY

NO. OF PAGES 1

DEPT	NAME		GROSS PAY		TRATE	
1 2	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	23 24 25	26 27 28 29 30 31	32 33 34 35 36 37	38 39	40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

TRATE = Tax Rate

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

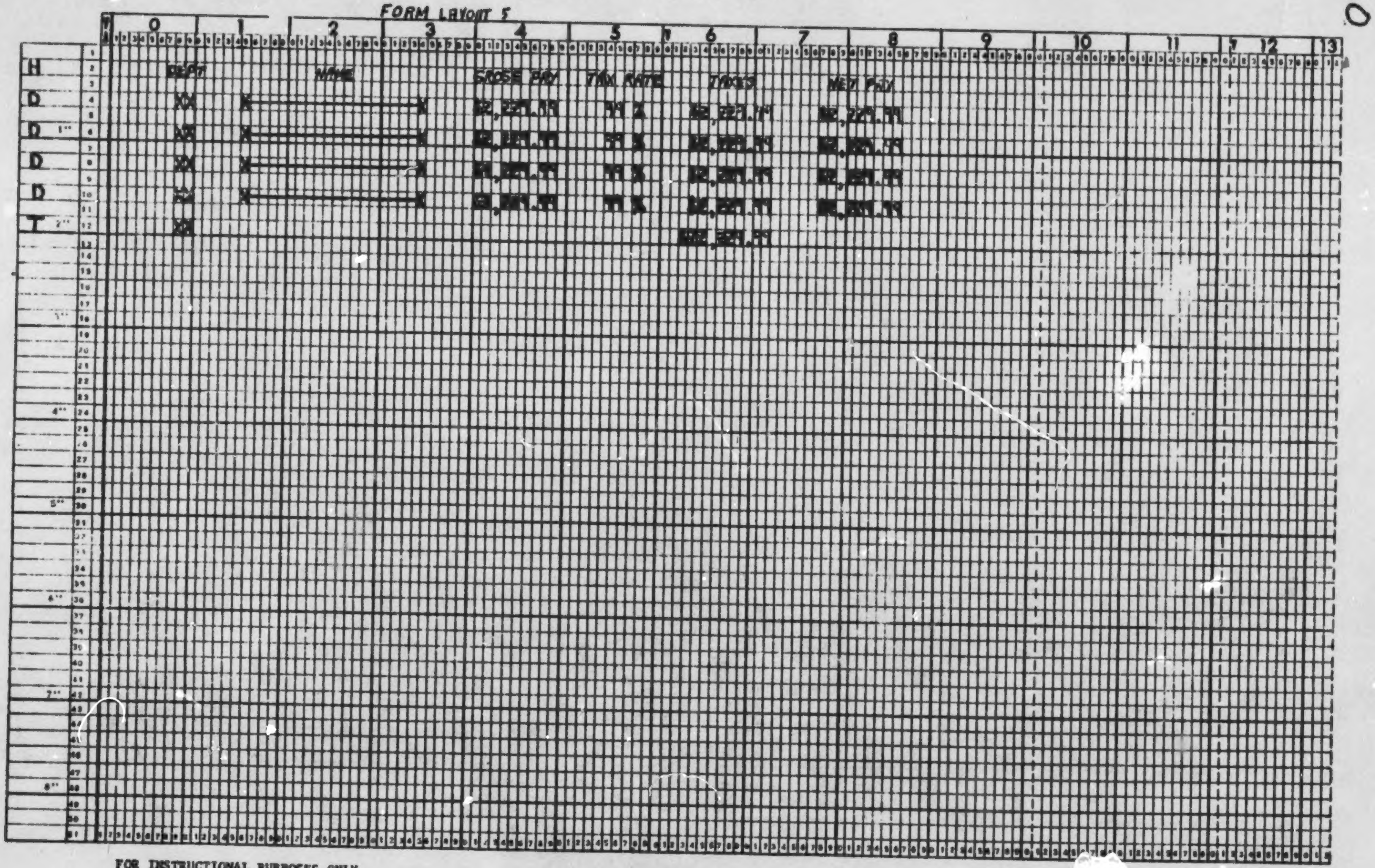
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

429

DA FORM 1 AUG 64 3165

U.S. GOVERNMENT PRINTING OFFICE: 1970 — 381-467

FORM LAYOUT


FORM LAYOUT 5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
H
D
D
D
D
T
1"
2"
3"
4"
5"
6"
7"
8"
DEPT
NAME
GROSS PAY
TAX RATE
TAXES
NET PAY
XX
X
X
ZZ,ZZ9.99
99 %
ZZ,ZZ9.99
ZZ,ZZ9.99
ZZZ,ZZ9.99

FOR INSTRUCTIONAL PURPOSES ONLY

X-221

ASSEMBLE A01
EXECUTE A02

ALC PE 7

PARTS INVENTORY

I. Introduction: The Fix or Repair Daily Corporation has decided to start selling by the case instead of breaking a case to sell individual items. They want you to write a program that will produce a report showing the number of full cases and the number of loose items for each stock number.

II. Definitions: Loose Items = the remainder of the division of

$$\frac{\text{Number of Items}}{\text{Number per Case}}$$

III. Input/Output:

A. Input: The input will be The Current Inventory File which is kept on cards (see card layout). **Device address = SYSIPT.**

B. Output: The output will be a printed report on the 1403 line printer (see the form layout for format and spacing). **Device address = SYSLST.**

IV. Assumptions: None.

V. Methods:

A. Provide main headings with a page number and column headings at the top of each page.

B. Edit all numeric fields to ensure that they contain numeric information.

1. If they do not, bypass all calculations and print out the card with fields in the appropriate positions. Insure that the Loose Items and Number of Cases fields are blank. The word "ERROR" should also be printed on the same line,

2. If they do, calculate the Number of Cases as follows:

$$\text{Number of Cases} = \frac{\text{Number of Items}}{\text{Number per Case}}$$

3. Print a detail line consisting of Stock Number, Item Name, Number of Items, Number of Cases, and Number of Loose Items editing as shown on the form layout.

C. Print only 20 double spaced detail lines per page.

VI. Edit Factors: Edit all numeric input fields to ensure that they contain numeric information.

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO. 1

NO. OF PAGES 1

SYSTEM CARD LAYOUT 6

PREPARED BY

432

	STOCK NUMBER	ITEM NAME	NUMBER OF ITEMS	I/C
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	53 54 55 56 57 58	59 60 61 62 63 64 65 66 67 68 69 70 71 72	73 74 75 76 77 78	79 80

I/C = Items per case

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

17

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

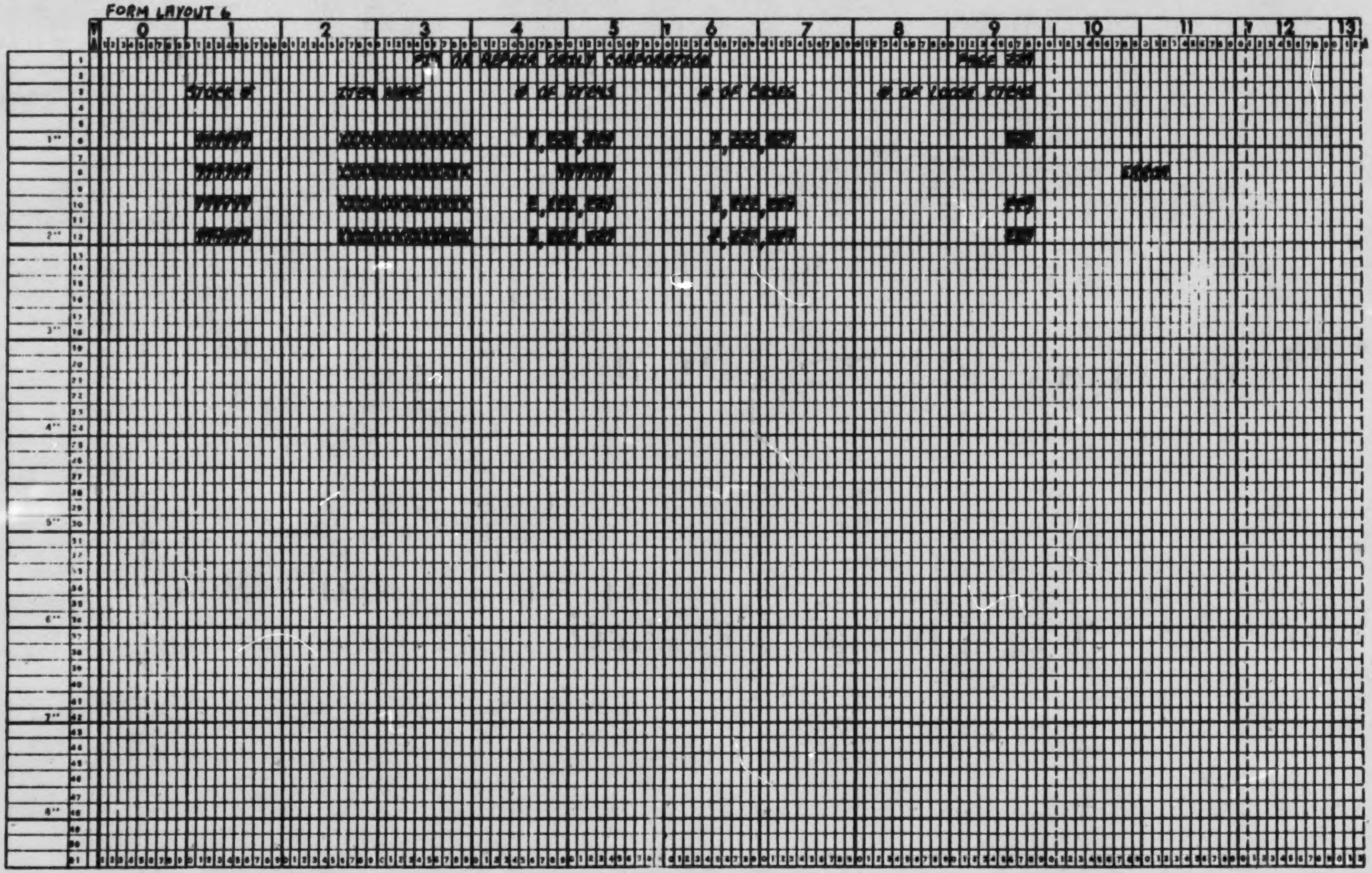
DA FORM 1 AUG 66 3165

U.S. GOVERNMENT PRINTING OFFICE 1970 — 361-987

492

491

FORM LAYOUT


FORM LAYOUT 6
STOCK #
ITEM NAME
# OF ITEMS
# OF CASES
# OF LOOSE ITEMS
1"
2"
3"
4"
5"
6"
7"
8"

FOR INSTRUCTIONAL PURPOSES ONLY

3:221

BLOOD TYPE

ALC PE 8

Assemble: A01
Execute: A05

You are required to determine the number of people with blood types A, B, AB, and O for a department in the Institute.

INPUT: 1) The input consists of two files: A <u>sequential disk</u> master file and a card file.

2) The format of the sequential disk records is given in on disk Layout 7. The file has a blocking factor of 3.

3) The card format is shown on Card Layout 7.

4) The blood type field might look like the following for example:

```
record position 31                         79

                ABCAABBOOCCCAAAABOCOOOAC...ABB
```

The blood type codes have the following meaning:

Blood Code	Blood Type
A	A
B	B
C	AB
O	O

Position 80 has a code. If it is anything other than the letter 'B', ignore that disk record.

5) The records are sorted by departments and one department may be on more than one record.

6) The card file contains one card with two numbers - a low key and a high key; these two numbers should be used to select the disk records to be processed. Only if the following condition exists should the disk record be processed:

low key ≤ disk record position 1-10 ≤ high key

If the condition does not exist, ignore that record.

OUTPUT: 1) The output will be on a 132 print position printer and consist of six lines for each organization.

2) Each organization should be on a separate page with the format given on Form Layout 7.

Assumptions: a) When the disk file was created, the name assigned to the file through Job Control was DISKIN. The device address is SYS006.

b) Ignore all illegal blood types found.

c) The totals can be expected to be at most 5 digits long.

d) Edit the totals by suppressing leading non-significant zeros.

436

TAPE LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Comptroller of the Army.

TAPE NO.	PREPARED BY	DATE	PAGE NO. 1
REMARKS Disk Layout 7			NO. OF PAGES 1

NO., LENGTH, BLOCK

80 X 1	Organization Number	seq	Organization Namee	1 to 49 Blood Type Code Entries
cont'd	Blood Type Entries (cond't)		code	

DA FORM 1 AUG 66 3166 USAFPP 294-63 10,?4 "FOR INSTRUCTIONAL PURPOSES ONLY"

D-500

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

SYSTEM Card Layout 7	PREPARED BY	PAGE NO. 1
		NO. OF PAGES 1

	Low Key	High Key	Not Used
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DA FORM AUG 66 3165

* U.S. GOVERNMENT PRINTING OFFICE: 1970 — 301-007

FORM LAYOUT

FORM LAYOUT 7

ORGANIZATION NUMBER 9———9

ORGANIZATION NAME X———X

TOTAL TYPE A 9—9

TOTAL TYPE B 9—9

TOTAL TYPE [illegible] 9—9

TOTAL TYPE C 9—9

22

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

437

ASSEMBLE A01
EXECUTE A06

ALC PE 9

MASTER FILE EDIT

I. Introduction: You are required to write a program that will edit a Transaction File and a Master File to ensure that they meet certain requirements. These files, if there are no edit errors, will be used as input for another program.

II. Definitions: None.

III. Input/Output:

Input: The Transaction File will be on cards (see card layout for specifications). The Master File will be on sequential disk. There are three records per block (see disk layout for specifications). **Card is on SYSIPT, Disk is on SYS006, File name for Disk is MASTIN.**
Output: A printed report will be produced to show the status of the different records (see form layout for specifications).
Device address = SYSLST.

IV. Assumptions: There will be one,and only one, transaction record per master record.

V. Methods:

1. Print a main heading with page number and column headings at the top of each page.

2. Print 15 double spaced detail lines per page.

3. Set up a one byte switch to be used when you edit the input.

4. Edit the classification field on the transaction card to find out whether it is blank, numeric, alphabetic, or alpha numeric. One of the first four bits of the switch in Step 3 above will be turned on to indicate what type of field it is.

5. Do the same thing for the classification field on the master record, except that one of the last four bits will be turned on to indicate what type of field it is.

6. The code on the transaction card will tell you what type this field should be.

CLASSIFICATION

TYPE OF CODE	TRANSACTION	MASTER
1	Blank	Blank
2	Anything	Numeric
3	Alphabetic	Not blank
4	Numeric	Numeric

7. If either or both of the fields are not the appropriate type, print the fields in the correct print positions along with the word "ERROR"

8. If both the fields are the appropriate type, print the fields in the correct print positions.

VI. Edit Factors: None

440

The following definitions of terms apply to ALC PE #4. These are further explained by the series of examples that follow the definitions.

1. BLANK - this means that all 10 positions of the field are blanks.
2. NON-BLANK - This means that at least 1 and maybe as many as all 10 positions of the field contain anything other than a blank.
3. NUMERIC - This means that all 10 positions of the field contain only the digits 0 thru 9.
4. ALPHABETIC - This means that all 10 positions of the field contains only the letters A thru Z, or blanks, however, all 10 positions may not be blanks.

EXAMPLES:

DATA FIELD	TYPE
ABCDEFGHIJ	ALPHABETIC, NON BLANK
12ƀƀ345678	NON BLANK
.(AXC6.*)/	NON BLANK
1234567890	NUMERIC, NON BLANK
ƀƀƀƀƀƀƀƀƀƀ	BLANK
ƀƀƀ642ƀƀƀƀ	NON BLANK
XXABCXX6XX	NON BLANK
2468013579	NUMERIC, NON BLANK
JOHNƀDOEƀƀ	ALPHABETIC, NON BLANK

503

MULTIPLE-CARD LAYOUT
(AR 18-7)

PAGE NO. 1

SYSTEM
TRANSACTION CARD CARD LAYOUT 8

PREPARED BY

NO. OF PAGES 1

CLASSIFICATION

CODE

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

26

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DA FORM 1 AUG 64 3165

DISK
[redacted] LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Comptroller of the Army.

TAPE NO.	
PREPARED BY	
DATE	
PAGE NO.	1
NO. OF PAGES	1
REMARKS	MASTER FILE DISK LAYOUT 8

NO., LENGTH, BLOCK

100 X 3	NOT USED (1–50)	
	NOT USED (51–90)	CLASSIFICATION (91–100)

27

442

DA FORM 1 AUG 66 3166

FORM LAYOUT

FORM LAYOUT 8

TITLE	1		MASTER EDIT REPORT		PAGE ZZZ9
	2				
HEADING	3	TRANSACTION	CODE	MASTER	
	4				
DETAIL	5	XXXXXXXXXX	X	XXXXXXXXXX	
LINES 1''	6				
	7	XXXXXXXXXX	X	XXXXXXXXXX	ERROR
	8				
	9	XXXXXXXXXX	X	XXXXXXXXXX	
	10				
	11				
2''	12				
	13				
	14				
	15				
	16				
	17				
3''	18				
	19				
	20				
	21				
	22				
	23				
4''	24				
	25				
	26				
	27				
	28				
	29				
5''	30				
	31				
	32				
	33				
	34				
	35				
6''	36				
	37				
	38				
	39				
	40				
	41				
7''	42				
	43				
	44				
	45				
	46				
	47				
8''	48				
	49				
	50				
	51				

28

443

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

444

ALC PE 10

ASSEMBLE A01
EXECUTE A09

BASE PAY EXERCISE

I. __Introduction:__ You are to write a program that will produce a one page report showing the current base pay for all enlisted personnel with less than 20 years service.

II. __Definitions:__ None.

III. __Input/Output:__

Input: None:

Output: The output will be on a 1403 line printer (SYSLST), and will consist of a one page report as shown on the attached Form Layout.

IV. __Assumptions:__ None.

V. __Methods:__

1. Print the column heading on the top of the page.
2. A macro called EMPAY has been catalogued on the system and it will generate the required nine pay tables, each table having eleven entries. Each entry of the tables will be a packed four byte field containing the old base pay. The label that you code on the macro will appear as the name of each table and will be followed by a number from one to nine which indicates which of the tables the label applies to. There are no operands required on the macro.
3. Each table will be in ascending order based on time in service.
4. Update each entry in all nine tables as follows:
   New Base Pay = Old Pay + Old Pay X 5.52%
5. When the tables have been updated print out the detail lines as shown on the Form Layout.
6. Edit the pay fields with a floating dollar sign, and a decimal pnt.
7. Use at least one BCT, BXLE, or BXH in your program.

VI. __Edit Factors:__ None.

FORM LAYOUT

FORM LAYOUT 9

	UNDER 2	OVER 2	OVER 3	OVER 4	OVER 6	OVER 8	OVER 10	OVER 12	OVER 14	OVER 16	OVER 18
E-1	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99
E-2											
E-3											
E-4											
E-5											
E-6											
E-7											
E-8											
E-9	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99	$ZZZZ9.99

EDIT ALL DOLLAR AMOUNTS WITH A FLOATING $

FOR INSTRUCTIONAL PURPOSES ONLY

3.321

445

446

ASSEMBLE A01
EXECUTE A02

ALC PE 11

BINARY MATH EXERCISE

I. Introduction: You have been interviewed for an ALC programming position by the Southeastern Box Company. They are interested in hiring you but prior to making their decision, they want you to submit a sample program. This company uses binary math for all calculations, so your sample program will demonstrate your ability in the use of binary math instructions. The firm also requires detailed source documentation on all ALC programs so this will be an additional requirement on this sample.

II. Definitions: None.

III. Input/Output:

A. Input: The input will be on cards (see attached card layout). **Device address = SYSIPT.**

B. Output: The output will be on the 1403 line printer (see attached form layout). **Device address= SYSLST.**

IV. Assumptions: None.

V. Methods:

A. Print a title and column headings at the top of each page.

B. Using a line counter, print only 20 double spaced detail lines per page.

C. Initialize a field called TOTAL at zero.

D. The indicator field on the input, when valid, will contain a one, two, or space. When it contains a one, bypass all ADD instructions until you encounter a two or a space in the indicator field.

E. The processing code from the input will determine what calculations should be performed on the number read from the card. If the processing code is a —

one - add the number to TOTAL;

two - Subtract the number from TOTAL

three - Divide the number into TOTAL keeping the quotient as the new TOTAL

four - Multiply the number times TOTAL

five - Add the number to TOTAL, then Multiply the new TOTAL by the number

513

six - Divide the number into TOTAL, then subtract the number from the new TOTAL

*** Any processing code that is larger than 6 is invalid.

F. Print out each card with the contents of TOTAL out to the right of the card.

G. If any errors are found on the card, bypass all calculations and print out the card, leaving the TOTAL field blank.

H. If the TOTAL is negative, edit a minus sign on the right side of the TOTAL field on the output.

VI. Edit Factors: Edit the card to insure that all data fields are numeric, except the indicator field which may be a space.

MULTIPLE CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

SYSTEM CARD LAYOUT 10	PREPARED BY	PAGE NO. 1 NO. OF PAGES 1

4448

NUMBER | P | I | NOT USED

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

P = Processing Code

I = Indicator

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

33

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

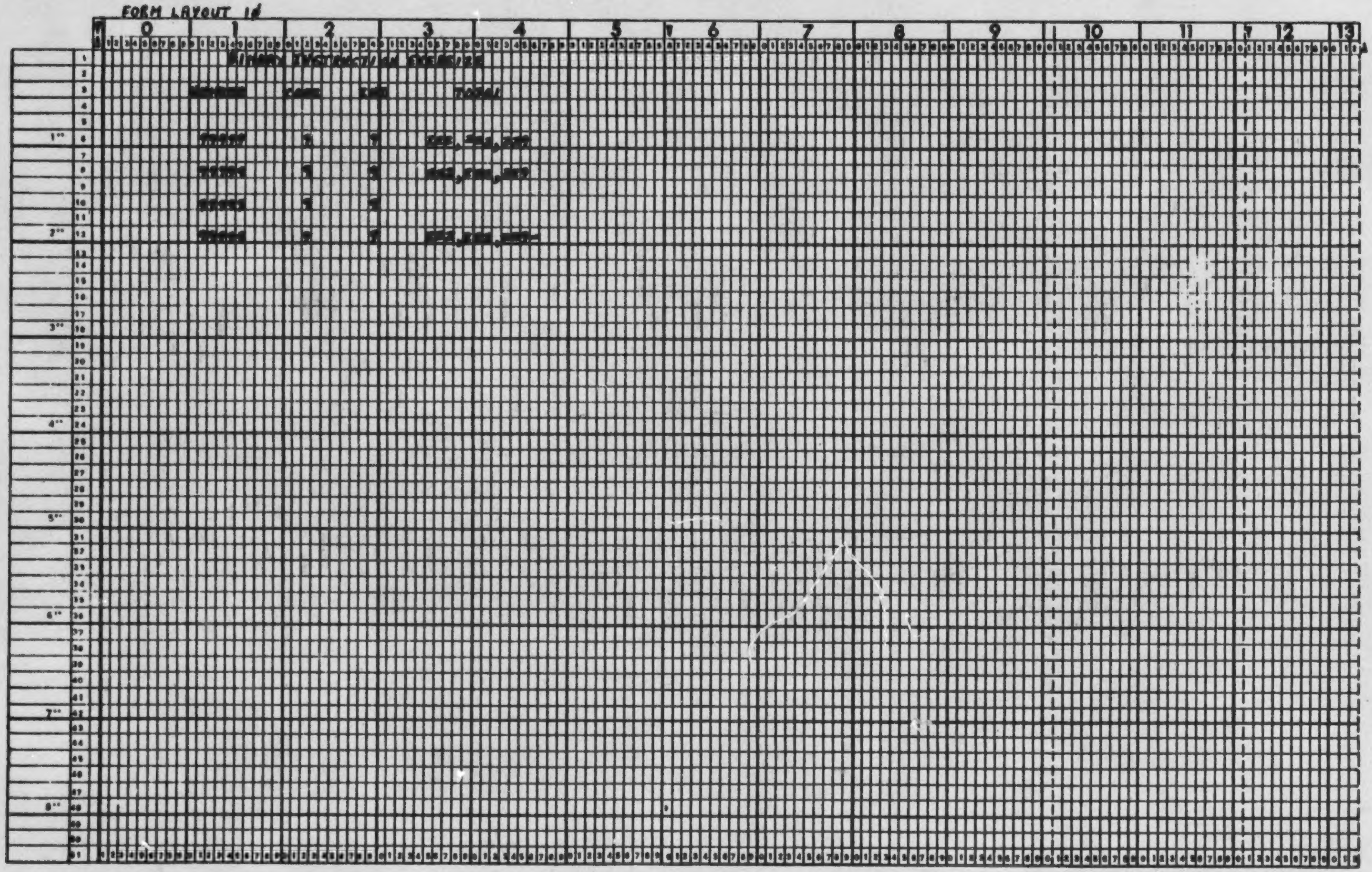
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DA FORM 1 AUG 66 3165

U.S. GOVERNMENT PRINTING OFFICE: 1970 – 381-667

FORM LAYOUT

FORM LAYOUT 16



BINARY INSTRUCTION EXERCISE

[illegible]	[illegible]	[illegible]	TOTAL
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

4449

450

INTEGER ARITHMETIC

ALC PE 12

Assemble: A01
Execute: A12

You are to write an ALC program that will determine which numbers in a given interval are perfect squares, and which numbers are prime. (Prime numbers are those numbers that are only divisible by 1 and themselves).

INPUT: The program will read one card. This card will contain the upper and lower bounds of the interval. The format of the card is shown on Card Layout 11.

OUTPUT: The output should start at the top of a new page. Detail lines should follow the appropriate headings.

Each detail line will consist of 16 8 character fields. Each field will contain one number, right justified. See Form Layout 11.

Useful Information:

1. If a number is a perfect square it is not a prime.
2. If a number other than 2 is even it cannot be a prime.
3. If checking a number N to see if it is prime, it is only necessary to check for divisibility by numbers $\leq \sqrt{N}$.

519

INSTRUCTIONAL USE ONLY

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

SYSTEM

CARD LAYOUT 11

PREPARED BY

NO. OF PAGES

Lower Bound	Upper Bound	

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

36

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

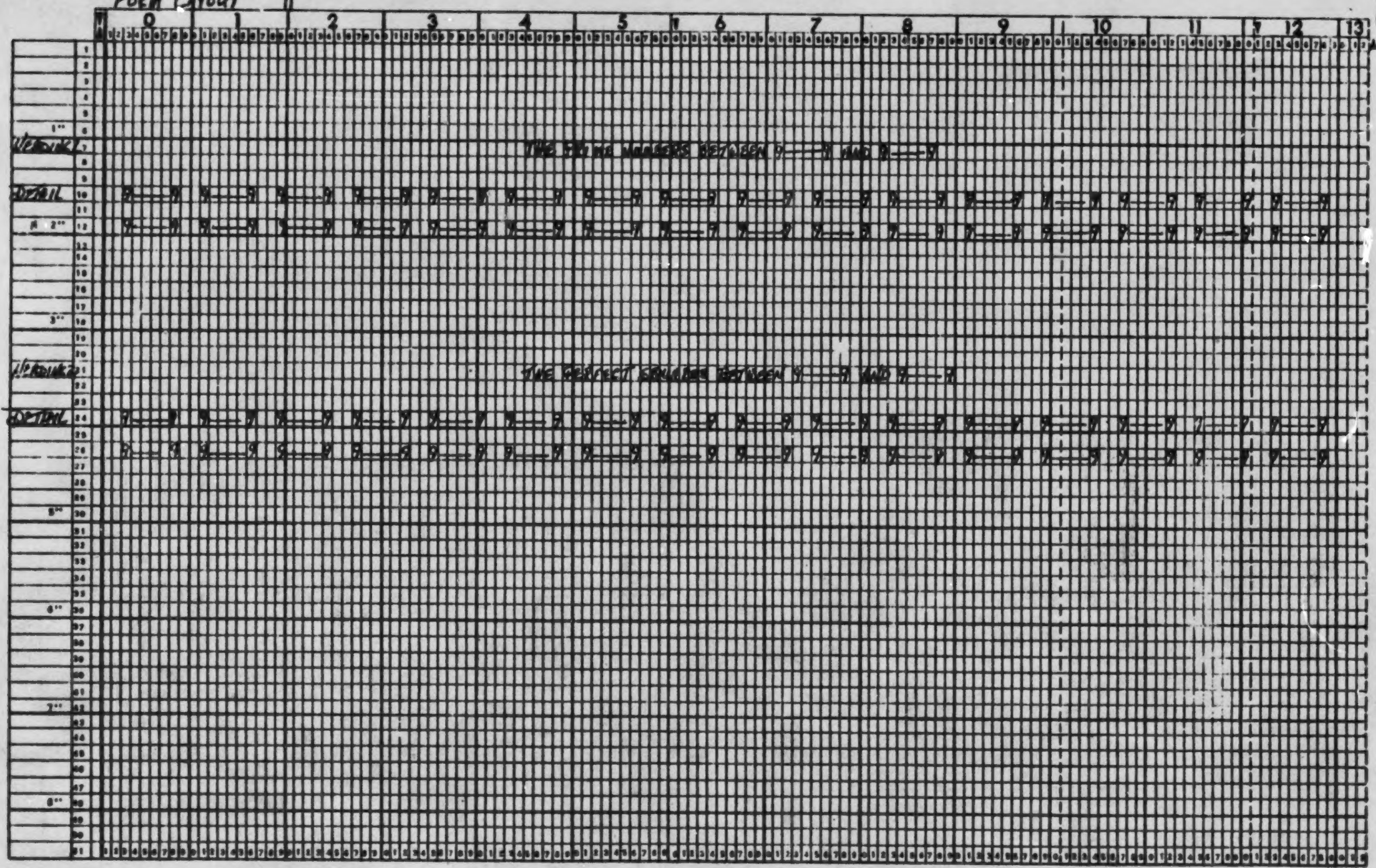
451

DA FORM 1 AUG 64 3165

U.S. GOVERNMENT PRINTING OFFICE: 1970 - 381-607

FORM LAYOUT

FORM LAYOUT



0
1
2
3
4
5
6
7
8
9
10
11
12
13
1"
2"
3"
5"
6"
7"
8"

HEADING — THE PRIME NUMBERS BETWEEN 9——9 AND 9——9

DETAIL — 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9

HEADING — THE PERFECT SQUARES BETWEEN 9——9 AND 9——9

DETAIL — 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9 9——9

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

453

Assemble A01
Execute A15

ALC PE 13

SORT

I. **Introduction**: The current personnel roster is maintained on disk as a sequential file, but due to additions the file is no longer in the proper order. You are to write an ALC program that will read the disk file, sort the records, and rewrite the records in ascending SSN order.

II. **Definitions**: None.

III. **Input/Output**:

A. Input: Your input is on sequential disk in the format shown on the attached disk layout. The filename for your input file is DISKINP and it is at device address SYS005.

B. Output: Your output will be on sequential disk and will be in the same format as it was on the input. The filename for output is DISKOPT and the device address is SYS006.

IV. **Assumptions**: There will be no more than 30 records on the file.

V. **Methods**: You may use any strategy you want to sort the records, but you must do your sort in core.

One method of sorting is the 'Bubble Sort'. To use this method first read all records into memory. Then take the first record and compare it to all subsequent records. If you find a SSN lower than the SSN you are using switch the two records and use the new SSN to continue the compares. At the end of the first pass you will have the record with the lowest SSN in the first position. Repeat the process beginning with the second record, then the 3rd and so forth until you have repeated it for all but the last record. At this point all of the records will be in the correct order.

EXAMPLE:

9, 6, 2, 3, 1, 4, 5, 7, 8

We would compare '9' to '6'. Since '6' is less than '9' we switch the '6' and the '9' and continue comparing with the '6'. Continuing with the '6' we find that '2' is less than '6' so once again we switch and continue this time with the '2'. At the end of the first pass the string would look like this: 1, 9, 6, 3, 2, 4, 5, 7, 8.

For the second pass we start with the second record which in this case is the '9' again. We compare it to the '6' (3rd record) switch and continue with the '6' and so forth. At the end of the second pass it would look like this: 1, 2, 9, 6, 3, 4, 5, 7, 8.

VI. **Edit Factors**: None.

VII. **Special Note**: The output disk will be dumped for you so that you may check your results. You need do nothing to cause this to happen.

454

DISK LAYOUT For use of this form, see AR 18-7; the proponent agency is Office of the Comptroller of the Army.	TAPE NO.	PREPARED BY	DATE	PAGE NO. 1
	REMARKS DISK LAYOUT 12			NO. OF PAGES 1

NO., LENGTH, BLOCK

80 x 1

ORG	SSN	NAME	PG*	RANK	ETS	DATE OF RANK	MON SER
1–3	4–12	13–30	31–32	33–35	36–41	42–47	48–50

------------MISCELLANEOUS INFORMATION -----------------------
51–80

* PAY GRADE

39

DA FORM 1 AUG 66 3166

526

525

FILL REPORT GENERATOR

ALC PE 14

Assemble: A01
Execute: A13

You are to write an ALC program that will generate a bar graph showing the percentage fill for all Federal Stock Numbers.

INPUT: Your program will read the fill data from an unlabeled tape. The tape may contain more than one entry for each FSN. The format of the tape is shown on Tape Layout 13.

The filename for the tape is TAPEIN and the Device address is SYS007.

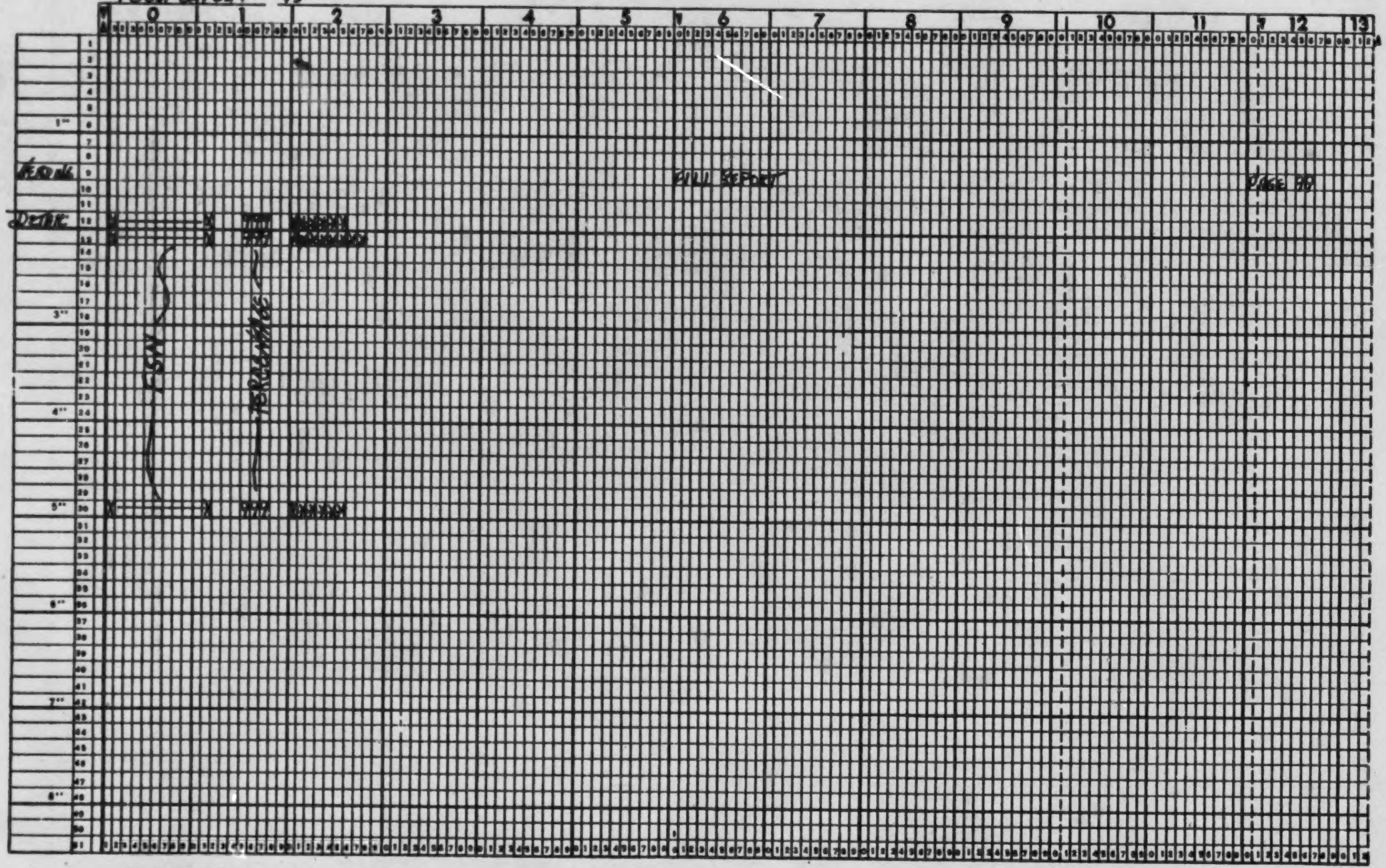
OUTPUT: The output will consist of a heading at the top of each page and a number of detail lines (one for each FSN). The detail lines will contain three fields; the FSN, the percentage fill, and a row of asterisks representing the percentage fill (one asterisk per percent). See Form Layout 13.

The device address is SYSLST.

Notes: You will have to watch for zero fill items and items which have not been ordered during the period. There is also the possibility that there may be errors on the tape and that the quantity filled will be greater than the quantity ordered. Print zero fill items as part of the graph. List the zero order items on a separate page, one per line under an appropriate heading (See Form Layout 13b). Also on this final page should be one line with the phrase 'LOWEST FILL PERCENTAGE WAS FOR', followed by the FSN of the item with the lowest fill percentage and the percentage. If there are two with the same percentage the one with the largest quantity ordered should be listed. If there are more of an item listed as filled than were ordered, list the FSN as part of the graph but instead of putting in a percentage put in the word 'ERROR'.

FORM LAYOUT

FORM LAYOUT 19



0
1
2
3
4
5
6
7
8
9
10
11
12
13
1"
3"
4"
5"
6"
7"
8"
HEADING
DETAIL
BILL REPORT
PAGE 99
FSN
YARDAGE

FOR INSTRUCTIONAL PURPOSES ONLY

3-221

42

530

529

457

TAPE LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Comptroller of the Army.

TAPE NO.	PREPARED BY	DATE	PAGE NO. 1
REMARKS Tape Layout 13			NO. OF PAGES 1

NO., LENGTH, BLOCK

NO., LENGTH, BLOCK	FSN	QTY Ordered	QTY Filled	
21 X 1	9 9	9 9	9 9	

DA FORM 1 AUG 66 **3166** USAFPP 294-65 10/74 "FOR INSTRUCTIONAL PURPOSES ONLY"

D-500

FORM LAYOUT

FORM LAYOUT 13b

	Line	
HEADING	8	FILL REPORT PAGE 99
DETAIL 2	11	X--FSN--X
	13	X--FSN--X
	23	X--FSN--X
DETAIL 3	26	LOWEST FILL PERCENTAGE WAS FOR X--FSN--X

FOR INSTRUCTIONAL PURPOSES ONLY

3-281

459

460

EDIT PROBLEM

ALC PE 15

Assemble: A01
Execute: A14

You are to write an ALC program to perform edit checks on cards that will be input to a requisition processing run.

INPUT: The format is as shown on Card Layout 14.

The fields should be edited as follows:

The FSN should be numeric (0-9).

The nomenclature may contain alphanumeric information (0-9 or A-Z) or blanks or dashes.

The unit of measure should be alphabetic (A-Z)

The quantity should be numeric (0-9).

The activity code should contain alphabetic information in the first two positions and numeric information in the last four.

OUTPUT: The heading is as shown on Form Layout 14.

The detail line should consist of the card image and the name of the column in error. Use FSN for federal stock number and NONE if no errors are found on the card.

The logical name for the printer is SYSLST, double space all output

INSTRUCTIONAL USE ONLY

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

SYSTEM

CARD LAYOUT 14

PREPARED BY

NO. OF PAGES

	Federal Stock Number		Nomenclature		UM		Quan		Actvty Code	
	1 2 3 4 5 6 7 8 9 10 11	12 13	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33	34 35	36 37 38	39	40 41 42 43 44	45 46 47 48 49	50 51 52 53 54 55	56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

46

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

461

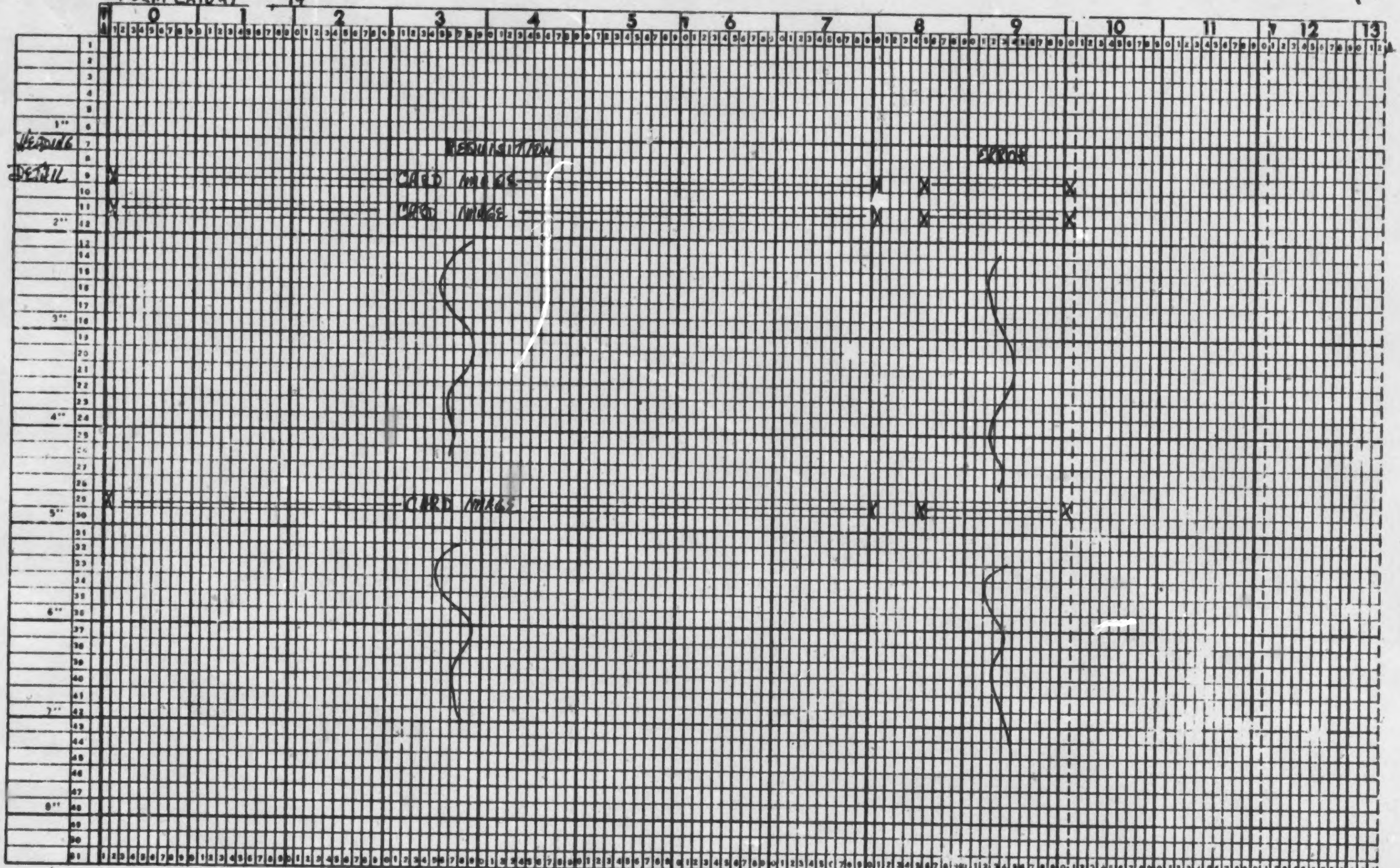
DA FORM AUG 64 3165

U.S. GOVERNMENT PRINTING OFFICE

462

FORM LAYOUT

FORM LAYOUT 14



HEADING 7 REQUISITION ERROR

DETAIL 9 X CARD IMAGE X X X

11 X CARD IMAGE X X X

29 X CARD IMAGE X X X

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

UNITED STATES ARMY
INSTITUTE OF ADMINISTRATION

MAY 1978

X

PERFORMANCE TEST

X

DEBUGGING A COBOL PROGRAM USING ALC

Name ______________________________

Student Number ______________ **Class Number** ________

INSTRUCTIONS:

1. **INDIVIDUAL WORK IS MANDATORY.** Appropriate disciplinary action will be taken against students who give or receive unauthorized help.
2. Special instructions are listed on page 1 of this booklet.
3. Insure that your class and student number are on the individual grade sheet and/or punchee cards.

3P/5M-9324

DEBUGGING COBOL PROGRAMS

Programs which do not go to a normal end of job are canceled by the operating system, and are considered to have abnormally terminated (ABEND). When this happens, page 1 of the program listing will contain, among other items, a system completion code (See Incl 1). The completion code is an indication of what caused the program to ABEND. Following is a list of the most commonly encountered system completion codes, a description of each, and an indication of the probably cause of the ABEND. Those causes marked with an asterisk (*) should be checked first.

COMPLETION CODE - 001.

a. Error Description.

Input/Output error.

b. Probable Causes.

(*) Wrong length record.

(*) Wrong length physical block.

No end of file marker.

(*) Attempt to read record after end-of-file condition found.

Physical damage to recording medium.

Device malfunction.

(*) Attempt to read from or write to a file which—

(1) Is closed.

(2) Was not opened.

COMPLETION CODE - 013.

a. Error Description.

Detection of conflicting or unsupported parameters during OPEN processing.

b. Probable Cause.

(*) Check for incorrect external name in a SELECT statement.

COMPLETION CODE - 031.

a. Error Description.

An input/output error was detected when processing under the indexed sequential access method (ISAM).

b. Probable Causes.

Physical damage to recording medium or device.

(*) Out of sequence key when loading an ISAM data set.

(*) Wrong length record or block.

COMPLETION CODE - 03B.

a. Error Description.

The error occurred during OPEN processing for an indexed sequential file (ISAM).

b. Probable Causes.

(*) The file had not been created.

The file had not been closed after creation.

COMPLETION CODE - 03D.

a. Error Description.

The error occurred during OPEN processing for an indexed sequential file (ISAM).

b. Probable Causes.

Indexed sequential organization not specified.

(*) Check for omission of "ACCESS IS RANDOM" clause in select statement for ISAM file.

COMPLETION CODE - OC1.

a. Error Description.

The operation code detected is not valid.

b. Probable Causes.

A branch to a data area; fetching of data as an operation code for this instruction.

(*) A missing or misspelled external name on SELECT statement.

(*) A file had not been opened when an input/output instruction was issued for it.

A file had been closed when an input/output instruction was issued for it (this may also cause an OC5 termination).

COMPLETION CODE - OC5.

a. Error Description.

An address is specified that is outside of the available storage of the particular computing system.

b. Probable Causes.

Invalid data address.

(*) Indexing (subscripting) outside the program's assigned limits.

(*) Uninitialized index (or subscript). This may also cause a data exception (OC7).

An input/output instruction triggered termination because OPEN was unable to complete its task.

A missing or misspelled external name in SELECT statement.

An attempt to CLOSE a file a second time.

(*) COBOL: An improper exit from a procedure being operated on by a PERFORM statement.

COBOL: A sort operation is being attempted with an incorrect cataloged procedure.

(*) COBOL: An attempt to reference an input/output area before READ or OPEN statement has been issued for the file.

COMPLETION CODE - OC7.

a. Error Description.

Data in a field was of incorrect format for the instruction attempting to process it.

b. Probable Causes.

(*) A data field was not initialized, e.g., blanks were read into a field designed to be processed with packed decimal instructions.

A packed decimal field had an incorrect sign field.

Uninitialized index or subscript. This may also cause an addressing (OC5) or protection (OC4) completion code.

Fields in decimal arithmetic overlap incorrectly.

The decimal multiplicant has too many high-order significant digits.

(*) The index (or subscript) value was incorrect and invalid data was referenced. This could also cause an addressing (OC5) or protection (OC4) completion code.

467

COBOL: Data was moved from the DISPLAY field to the COMPUTATIONAL or COMPUTATIONAL-3 field at group level. No conversion was performed; invalid data for COMPUTATIONAL-3 (packed decimal) results.

COBOL: The figurative constants ZERO or LOW-VALUE was moved to a group level numeric field.

COBOL: Omission of USAGE clause or inclusion of an erroneous USAGE clause.

FOR USAIA USE: (See Incl 1)

a. Extract Program Interruption Address (PIA) from page 1 of program listing.

b. Determine entry point (EPA) for program from page 1 of program listing. It will be either—

(1) 01C020.

(2) 034020.

c. Subtract EPA from PIA. The result is a displacement. Remember, all addresses are in hexidecimal as is the displacement. Now go to the page in the listing which has the heading 'CONDENSED LISTING'. Each verb used in the source program is listed and has associated with it, a statement number and a displacement. To find the statement which caused the program to ABEND, locate the displacement on the condensed listing which most nearly matches and is less than the displacement you just calculated. The corresponding statement number is the one which caused the problem. Now, go back to the probable cause paragraph and correct problem.

COMPLETION CODE - OC9 OR OCB.

a. Error Description.

Divide exception.

b. Probable Cause.

Attempting to divide by zero.

COMPLETION CODE - D37.

a. Error Description.

Space allocated for a data set on a direct access device was exceeded.

b. Probable Cause.

The program may be in a loop and is writing an infinite number of records to an output file.

468

COMPLETION CODE - 222.

a. Error Description.

Operator canceled the job.

b. Probable Causes.

The job may have been canceled because it appeared to be in a loop or because it was waiting for resources that were not immediately available. Perhaps the job was canceled to correct a system interlock condition. There are many reasons why an operator might cancel a job. There may be nothing wrong with your program.

COMPLETION CODE - 322.

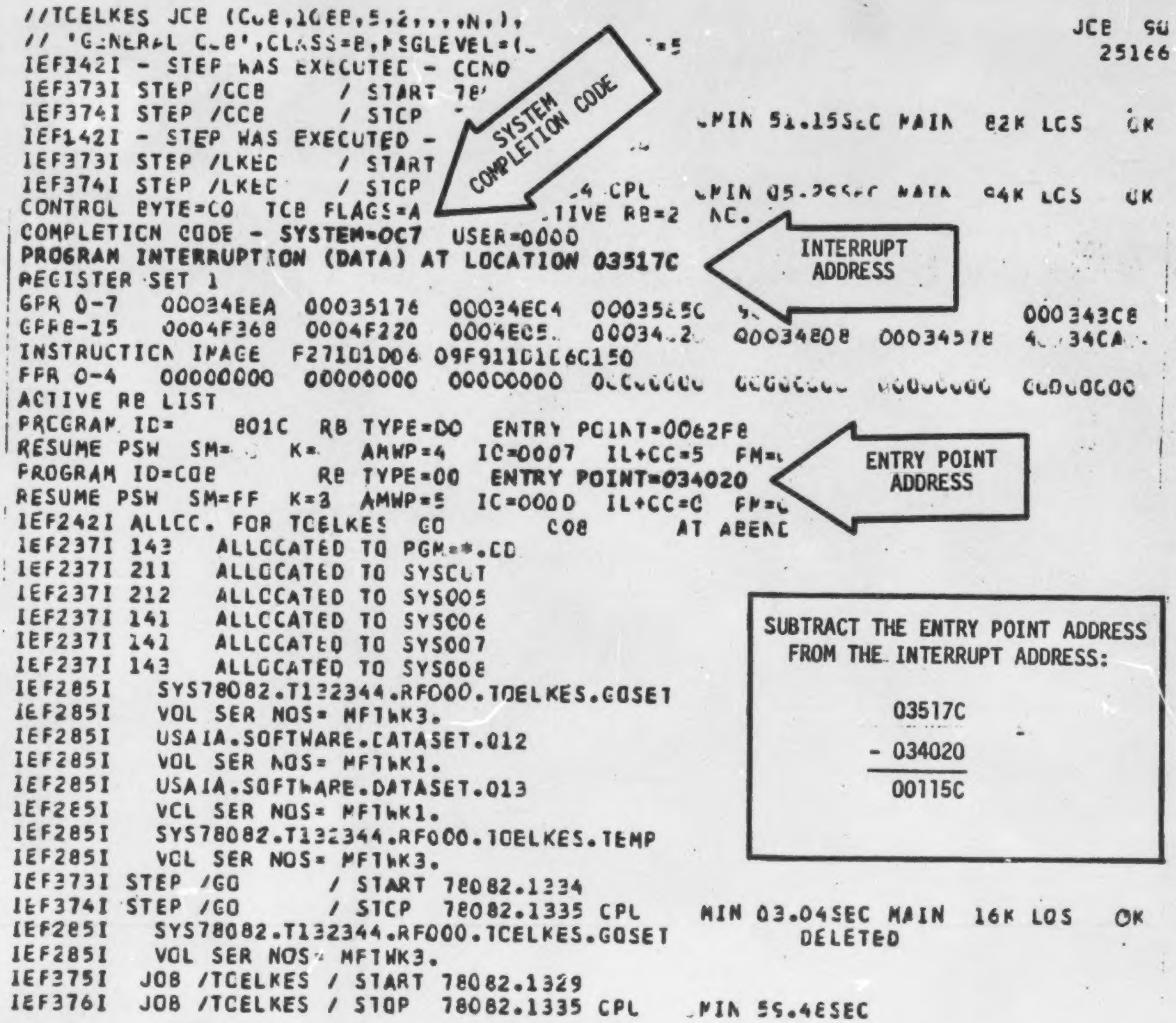
a. Error Description.

Execution of a program took longer than the time allocated.

b. Probable Cause.

Program is in a loop.

469


SYSTEM COMPLETION CODE
INTERRUPT ADDRESS
ENTRY POINT ADDRESS

```
//TCELKES JOB (C08,1088,5,2,,,,N,),                                              JOB  50
// 'GENERAL C08',CLASS=B,MSGLEVEL=(           =5                                 25166
IEF142I - STEP WAS EXECUTED - COND
IEF373I STEP /COB      / START 78
IEF374I STEP /COB      / STOP                           MIN 51.15SEC MAIN  82K LCS   OK
IEF142I - STEP WAS EXECUTED -
IEF373I STEP /LKED     / START
IEF374I STEP /LKED     / STOP               .4 CPU    MIN 05.25SEC MAIN  94K LCS   OK
CONTROL BYTE=C0  TCB FLAGS=A             TIVE RB=2  NC.
COMPLETION CODE - SYSTEM=0C7  USER=0000
PROGRAM INTERRUPTION (DATA) AT LOCATION 03517C
REGISTER SET 1
GPR 0-7    00034EEA  00035176  00034EC4  000358 50  9                       000343C8
GPR8-15    0004F368  0004F220  0004EC5   00034 2    00034808  00034578  4  34CA
INSTRUCTION IMAGE  F27101D06 09F911D1C6C150
FPR 0-4    00000000  00000000  00000000  00000000  00000000  00000000  00000000
ACTIVE RB LIST
PROGRAM ID=    801C  RB TYPE=D0  ENTRY POINT=0062F8
RESUME PSW  SM=    K=    AMWP=4  IC=0007  IL+CC=5  FM=
PROGRAM ID=COB        RB TYPE=00  ENTRY POINT=034020
RESUME PSW  SM=FF  K=3  AMWP=5  IC=000D  IL+CC=C  FM=
IEF242I ALLOC. FOR TCELKES  GO          C08       AT ABEND
IEF237I 143   ALLOCATED TO PGM=*.DD
IEF237I 211   ALLOCATED TO SYSOUT
IEF237I 212   ALLOCATED TO SYS005
IEF237I 141   ALLOCATED TO SYS006
IEF237I 141   ALLOCATED TO SYS007
IEF237I 143   ALLOCATED TO SYS008
IEF285I    SYS78082.T132344.RF000.TOELKES.GOSET
IEF285I    VOL SER NOS= MFTWK3.
IEF285I    USAIA.SOFTWARE.DATASET.012
IEF285I    VOL SER NOS= MFTWK1.
IEF285I    USAIA.SOFTWARE.DATASET.013
IEF285I    VOL SER NOS= MFTWK1.
IEF285I    SYS78082.T132344.RF000.TOELKES.TEMP
IEF285I    VOL SER NOS= MFTWK3.
IEF373I STEP /GO       / START 78082.1334
IEF374I STEP /GO       / STOP  78082.1335 CPU    MIN 03.04SEC MAIN  16K LOS   OK
IEF285I    SYS78082.T132344.RF000.TCELKES.GOSET             DELETED
IEF285I    VOL SER NOS= MFTWK3.
IEF375I  JOB /TCELKES / START 78082.1329
IEF376I  JOB /TCELKES / STOP  78082.1335 CPU    MIN 59.48SEC
```

SUBTRACT THE ENTRY POINT ADDRESS
FROM THE INTERRUPT ADDRESS:

$$\begin{array}{r} 03517C \\ -\ 034020 \\ \hline 00115C \end{array}$$

1

```
                                                        CONDENSED LISTING
     195          OPEN          000902                 197         PERFORM      000A40
     201          OPEN          00084E                 202         MOVE         000EE6
     205          MOVE          0008A0                 206         MOVE         000EAA
     209          CLOSE         0008DA                 212         STOP         000C56
     215          MOVE          00006B                 216         READ         000C6E
     219          IF            000094                 220         MOVE         000CA0
     223          MOVE          000006                 224         IF           000CCC
     227          GO            000CEA                 228         IF           000CF0
     230          GO            000D1A                 231         IF           000D20
     233          GO            000D4A                 234         IF           000D50
     236          GO            000D7A                 237         IF           000D60
     239          GO            000DAA                 240         PERFORM      000D80
     244          READ          000DD4                 246         MOVE         000DF4
     248          PERFORM       000E00                 250         EXIT         000E1E
     253          WRITE         000E2A                 255         EXIT         000E68
     258          IF            000E74                 259         PERFORM      000E80
     261          ELSE          000EA4                 261         MOVE         000EA4
     263          MOVE          000EAE                 264         PERFORM      000EB4
     267          EXIT          000FDA                 269         MOVE         000F10
     272          IF            000F22                 273         MOVE         000F38
     275          MOVE          000F54                 276         IF           000F5A
     278          IF            000F76                 279         MOVE         000F8C
     281          MOVE          000F90                 282         IF           000FA2
     284          IF            000FD6                 285         MOVE         000FE4
     287          MOV                                  288         MOVE         00100E
     290                                               292         EXIT         001074
     296                                               297         MOVE         00108C
     300                                               302         MOVE         0010B6
     304                        001002                 305         IF           0010CC
     307                        0010EE                 308         MOVE         001100
     310          MOVE          001122                 311         PERFORM      001132
     315          IF            001156                 316         WRITE        001168
     322          MOVE          001108                 323         MOVE         0011CE
     325          MOVE          001182                 326         MOVE         0011EC
     328          MOVE          0011FE                 329         MOVE         0011FE
     331          IF            00120A                 332         MOVE         001228
     334          MOVE          001268                 335         IF           0012EA
     337          WRITE         0012CA                 340         ADD          001306

 *STATISTICS*       SOURCE RECORDS =    342      DATA DIVISION STATEMENTS =    101
 *OPTIONS IN EFFECT*       SIZE =    86 16  BUF =     8192  LINECNT = 57  SPACE1,
 *OPTIONS IN EFFECT*      NOCMAP, NOPMAP,   CLIST,    SUPMAP, NOXREF,    LOAD, NC
 *OPTIONS IN EFFECT*         ZWB
```

USING THE CONDENSED LISTING, DETERMINE THE COBOL STATEMENT CAUSING THE INTERRUPT.

471

```
               EXIT.


           0060-DELETE.
               MOVE 'CNN' TO WS-OM-READ-SW,
                   WS-TRANS-READ-SW.
               MOVE IM01-EM-RCD TO OM01-NEM-RCD.
               MOVE 'DELETION' TO WS-DL-STATUS.
               PERFORM 0080-WRITE THRU 0080-EXIT.
           0060-EXIT.
               EXIT.


           0070-ERROR.
               MOVE 'CNN' TO WS-TRANS-READ-SW.
               MOVE IT01-TRANS-RCD TO OM01-NEM-RCD.
               MOVE 'BAD UPD' TO WS-DL-STATUS.
               IF IT01-PAY-TEST EQUAL SPACES
                   MOVE IT01-PAY-TEST TO WS-DL-PAY-ERR.
               IF IT01-FICA-TEST EQUAL SPACES
                   MOVE IT01-FICA-TEST TO WS-DL-FICA-ERR.
               IF IT01-IRS-TEST EQUAL SPACES
                   MOVE IT01-IRS-TEST TO WS-DL-IRS-ERR.
               PERFORM 0080-WRITE THRU 0080-EXIT.
           0070-EXIT.
               EXIT.
```

POSSIBLE CAUSE: WS-LINE-COUNT IS NOT INITIALIZED. CHECK DATA DIVISION.

```
           0080-WRITE.
               IF WS-LINE-COUNT GREATER THAN 19
               WRITE CR01-PRINT-RCD FROM
                   WS-READING-LINE-1
                   AFTER ADVANCING CHAN1
               WRITE CR01-PRINT-RCD FROM
                   WS-READING-LINE-2
                   AFTER ADVANCING 2 LINES
               MOVE ZERO TO WS-LINE-COUNT.
               MOVE OM01-SSN1 TO WS-DL-SSN1.
               MOVE OM01-SSN2 TO WS-DL-SSN2.
               MOVE OM01-SSN3 TO WS-DL-SSN3.
               MOVE OM01-NAME TO WS-DL-NAME.
               MOVE OM01-IN TO WS-DL-IN.
               MOVE OM01-ST-ADD TO WS-DL-ST-ADD.
               MOVE OM01-CITY TO WS-DL-CITY.
               MOVE OM01-JC-CODE TO WS-DL-JC-CODE.
               IF OM01-PAY      NUMERIC
               MOVE OM01-PAY TO WS-DL-PAY.
               IF OM01-FICA     NUMERIC
               MOVE OM01-FICA TO WS-DL-FICA.
               IF OM01-IRS      NUMERIC
               MOVE OM01-IRS TO WS-DL-IRS.
               WRITE CR01-PRINT-RCD FROM
                   WS-DETAIL-LINE
                   AFTER ADVANCING 2 LINES.
```

3

```
00087             05  FILLER                          PIC X(1).

00088        FD   OM-NEM-FILE
00089             BLOCK CONTAINS 5 RECORDS,
00090             RECORD CONTAINS 80 CHARACTERS,
00091             LABEL RECORDS ARE STANDARD,
00092             DATA RECORD IS OM01-NEM-RCD.
00093        01   OM 1-NEM-RCD.
00094             05  OM01-SSN.
00095                 10  OM01-SSN1
00096                 10  OM01-SSN2                   PIC 9(3).
00097                 10  OM01-SSN3                   PIC 9(2).
00098             05  OM01-NAME                       PIC 9(4).
00099             05  OM01-IN                         PIC X(14).
00100             05  OM01-ST-ADD                     PIC X(2).
00101             05  OM01-CITY                       PIC X(29).
00102             05  OM01-JC-CODE                    PIC X(14).
00103             05  OM01-PAY                        PIC X(1).
00104             05  OM01-FICA                       PIC S9(5)V9(2).
00105             05  OM01-IRS                        PIC S9(3)V9(2).
00106             05  FILLER                          PIC S9(5)V9(2).
                                                      PIC X(1).

             WORKING-STORAGE SECTION.
             77   WS-TRANS-READ-SW                    PIC X(3)
                                VALUE 'ONN'.
00110        77   WS-CM-READ-SW                       PIC X(3)
00111                           VALUE 'ONN'.
00112        77   WS-READ-COM-SW                      PIC X(3)
00113                           VALUE 'OFF'.
00114        77   WS-LINE-COUNT                       PIC 9(2).
00115        77   WS-NINES                            PIC 9(9)
00116                           VALUE 999999999.

00117        01   WS-PEADING-LINE-1.
00118             05  FILLER                          PIC X(54)
00119                           VALUE SPACE.
00120             05  WS-PO1-TITLE                    PIC X(24)
00121                           VALUE 'DAC MASTER UPDATE REPORT'.
00122             05  FILLER                          PIC X(55)
00123                           VALUE SPACE.

00124        01   WS-READING-LINE-1.
00125             05  FILLER                          PIC X(5)
00126                           VALUE SPACE.
00127             05  FILLER                          PIC X(12)
00128                           VALUE 'STATES'.
00129             05  FILLER                          PIC X(18)
00130                           VALUE 'SOC-SEC-NUM'.
00131             05  FILLER                          PIC X(13)
```

473

US ARMY INSTITUTE OF ADMINISTRATION
Fort Benjamin Harrison, Indiana 46216

OPERATING SYSTEMS

PRAECIPERE MILITI

Prepared for: ADPOC, NCOAC, P/AC

Prepared by: Software Division, Computer Science Department

SEPTEMBER 1978

STUDENT NAME: ______________________________

CLASS NO. ________________

FOR INSTRUCTIONAL PURPOSES ONLY

550

IA-01-06-24-25-26

TABLE OF CONTENTS

PAGE

Introduction 1-1
Disk Operating System (DOS) 1-4.5
Initial Program Load (IPL) 1-5
Multiprogramming 1-5.4

Job Control Language 3-1
Compile/linkedit/execute 3-4
JOB card 3-5
OPTION card 3-5

USAIA standard options 3-10
EXEC card 3-11
Linkage Editor 4-1
PHASE card 4-1

INCLUDE card 4-2
ENTRY card 4-2
ACTION card 4-3
Linkedit map 4-3

Assingment of I/O devices 5-1
ASSGN card 5-4
ADAM 5-4.2
Data file specification 6-1

Tape labels 6-1
TLBL card 6-3
LBLTYP card 6-5
Disk files 7-1

DLBL card 7-2
EXTENT card 7-2
COBOL PE2/PE3 JCL exercise 7-7
Indexed Sequential (ISAM) files 7-8

Librarian 8-1
Service (list/punch) functions 8-3
Catalog functions 8-13
Rename Functions 8-17

Delete Functions 8-18
Condense functions 8-20
Update functions 8-21
Reallocate functions 8-25

Copy functions 8-26
DOS utilities 9-1
File-To-File Utilities 9-5
Utility control cards 9-7

TABLE OF CONTENTS (CONT)

Operating Systems (OS) 10-1

APPENDICIES:

DOS Job Control flow chart A
Compile/Linkedit/Execute diagram B

476

HISTORY OF COMPUTERS:

The first type of computing machines that were used were the punch card machines. These machines used plugboards to store their programs. They were comparatively slow, had very limited capabilities and had no internal storage.

The PCM were followed by the first generation computers. These were big, relatively slow vacuum tube machines. All programming for these machines was done at the machine language level. They were single job initiation machines and were used mainly as scientific calculators.

About 1955-56, the second generation machines emerged. These had transistors instead of tubes and were faster than their first generation predecessors. It was during the second generation that complier level languages such as COBOL and FORTRAN were developed. The second generation machines had a batch job capability and were the first of the general purpose machines. They were now used for Business Data Processing as well as scientific calculators.

Time marches on....about 1961, enter the third generation machines replacing the transistor with semiconductor circuits once again increasing the speed. Magnetic core was also being produced relatively cheaply so these computers had large core memories.

All of these changes did not come without some problems, most of which started in the second generation. The hardware technology was ahead of the software support so there was an inefficient use of the hardware. Large computers were not effective. Many of their resources such as memory and I/O devices sat idle a lot of the time. There was too much human intervention; too much interaction between man and machine. Even though there was too much interaction between man and machine, there was no way for man to interact with man thru the machine, in other words, there was no way to link one program to another.

The solution to these problems was an operating system. An operating system is an integrated set of programs designed to improve overall operating effectiveness. The purpose of an operating system is to increase the amount of data processed in a given amount of time. An operating system will not reduce the execution time for a single program. Over the years there have been many operating systems. One of the first was the Share Operating System (SOS) written by a group of IBM users. The SOS did not pan out. IBM produced IBSYS for its 7090 line of computers. This was about 1959. In 1961, Burroughs introduced the Master Control Program (MCP). It was the first of the third generation operating systems. It was not, however, until about 1963 when IBM announced DOS and OS that operating systems were really here to stay. There are three basic sources for operating systems. The first and least common is the user group, the time and expense (about $60,000,000 for DOS) makes it almost impossible for a group of users to produce their own system. The second source is the Academic World. Universities often write their own special application systems using the wealth of research personnel on their faculties and any free graduate students. The last and most common source of operating system is the hardware manufacturer. Almost all operating systems come from this last source. There are three basic requirements for an operating system. The first is a control program to control all the systems' resources. The second is a language for man-system communication and finally a software library and some means for retrieving programs from that library. Not all operating systems will have all of these requirements in a very obvious form but these three needs will in some manner be present in every operating system.

Reduction in Costs: One of the first computers designed and built costs $500,000, had 18,000 vacuum tubes, weighted 30 tons and was named "ENIAC". Today, a microprocessor can have the same computing power of ENIAC, can be the size of a baby's fingernail and cost $10.

478

TYPES OF OPERATING SYSTEMS:

BPS (Basic Programming Support) - a minimum system resident in cards or tape for use in installations having no disk files. Support Assembler Language and RPG.

BOS (Basic Operating System) - A disk resident system supporting Assembler Language and RPG, with sufficient capacity to satisfy requirements of small installations.

TOS (Tape Operating System) - A tape resident system offering the capabilities of supporting all languages plus multiprogramming, storage protection, and the internal timer with no disk.

DOS (Disk Operating System) - a disk resident system for users requiring more advanced operational support than that of TOS. Supports all languages as well as remote processing, storage protection, multiprogramming, and the interval timer.

OS (Operating system) - a disk resident system that provides for the assignment of devices and auxillery storage space at execution time. Provides for multiprogramming fifteen seperate jobs at one time. Also provides an internal SPOOLing capability as well as a priority scheduling system.

OS/VS (Operating System / Virtual Storage) - a disk resident system that uses a Direct access storage device to simulate a larger memory capacity than the machine really has. For example a System 370/165 with 512,000 bytes of real memory can be made to perform as though it had 2,000,000 bytes of real memory.

OPERATING SYSTEM.

An Operating System - intergrated set of programs designed to improve the total operating efficiency of a computer's operation.

Functions of the Operating System.

- Scheduling of I/O operations
- Protection of partition boundaries (required for multiprogramming)
- Error handling as referenced to the five interrupts.

- Machine check interrupt
- Supervisor call (SVC) interrupt
- Program check interrupt
- External interrupt
- I/O interrupt

Operator communication with CPU.

Provide for program loading from private or system core image library.

Provide for program termination.

REQUIREMENTS OF THE OPERATING SYSTEM.

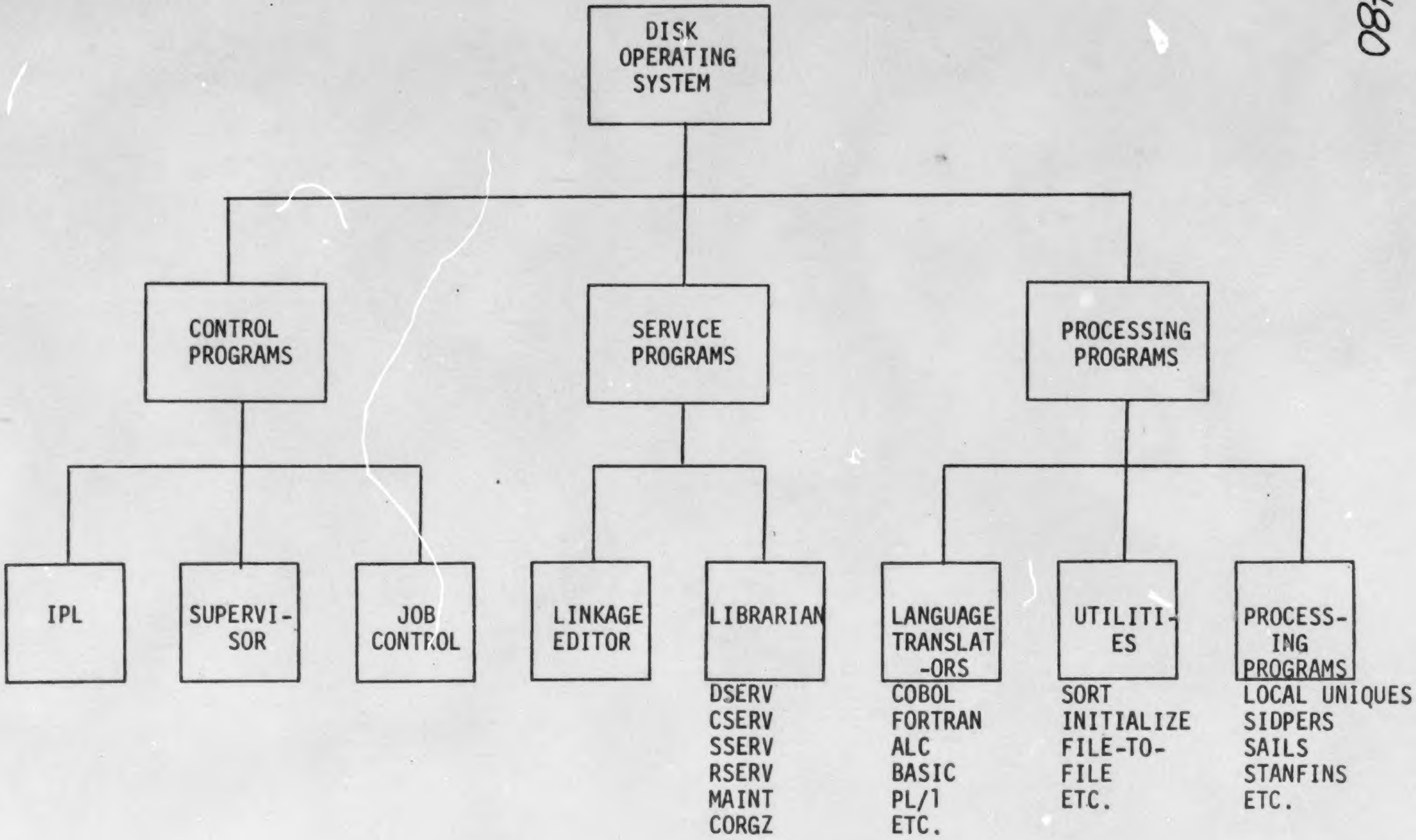
A control program (supervisor)

Library facilities

- Core image library
- Source statement library
- Relocatable library.

A device which provides a means for communication between the operating system and the computer operator (console typewriter).

NOTE: These libraries will be present in the operating system in some form.


DISK OPERATING SYSTEM
CONTROL PROGRAMS
SERVICE PROGRAMS
PROCESSING PROGRAMS
IPL
SUPERVISOR
JOB CONTROL
LINKAGE EDITOR
LIBRARIAN
DSERV
CSERV
SSERV
RSERV
MAINT
CORGZ
LANGUAGE TRANSLATORS
COBOL
FORTRAN
ALC
BASIC
PL/1
ETC.
UTILITIES
SORT
INITIALIZE
FILE-TO-FILE
ETC.
PROCESSING PROGRAMS
LOCAL UNIQUES
SIDPERS
SAILS
STANFINS
ETC.

V. Types of programs which make-up the operating system.

a. Initial Program Load (IPL)

INITIAL PROGRAM LOAD

MAIN MEMORY
SYS RES
IPL
CIL Supervisor
RL
SSL

IPL - is loaded by the operator dialing the address of SYS RES on the CPU and pressing the console load key. The loading of IPL is a machine function, no instructions are executed. After the IPL is loaded, it is given control of the machine. At this time IPL clears main memory and the registers, loads the supervisor in the low core of main memory and passes control to the supervisor. When these tasks are accomplished, IPL has no further usage.

AT The time when the supervisor is given control of the machine, the computer operator sets the date and time of day to be used by the supervisor and Problem Programs. He may also add or change the I/O device assignments in the Logical Unit Block (LUB) and Physical Unit Block (PUB) of the supervisor. Finally, he can change the partition sizes of the partitions allocated to the machine by entering an ALLOC statement.

EXAMPLE:

```
OIIOA  Give IPL Control commands
SET DATE = DD/MM/YY, clock=HH/MM/SS
OI20I DOS IPL Complete
```

At this point control is given to the control program (Supervisor)

The supervisor loads the Job Control Program into the Background partition. Job control finds the standard assignment for SYSRDR in the LUB and PUB. Next it read the job control statements from the card reader (SYSRDR) performs the following functions:

1. Assigns device addresses to symbolic units.
2. Set up fields in the communication region.
3. Edit and stores label information.
4. Prepares for restarting checkpoints programs.
5. Clear the problem area to binary zeros between job steps.
6. Finally, it passes the name of the program to be executed to the System Loader.

7. Control is then passed to the Supervisor.

When the supervisor is given control from the Job Control Program, the System Loader goes to System Residence Pack (SYSRES), Core Image Library and fetches the program to be executed by placing it into main memory. Next the address of the first executable instruction is stored in the Control Unit of the CPU.

Finally, the problem program is given control and begins execution.

The preceding process explains what happens in the computer to start a problem program executing in the Background (BG) partition.

In order to start jobs in the Foreground 1 (F1) and Foreground 2 (F2) partitions, the operator must press the request key on the console typewriter and enter the following statements:

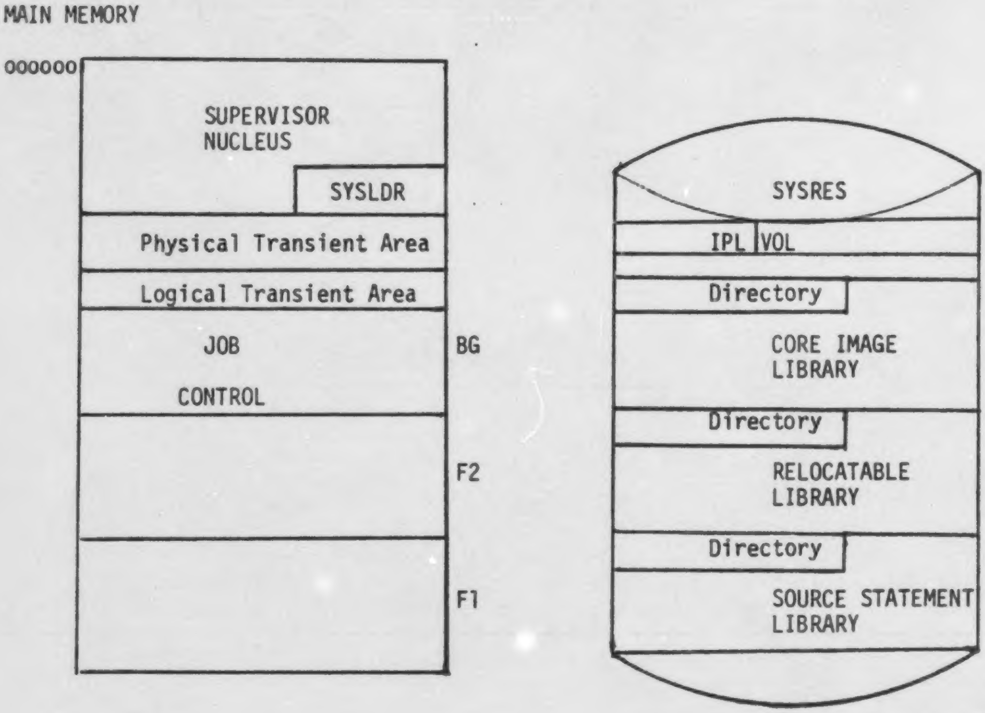
```
AR   BATCH F1
FI   ASSGN  SYSXXX,X'CUU'
PRESS THE REQUEST KEY
AR   BATCH F2
F2   ASSGN SYSXXX,X'CUU'
```

This process caused the Job Control Program to be loaded in the F1 and F2 partitions. Once Job Control is given control, it repeats the steps indicated above which cause the problem program to begin executing in each partition.

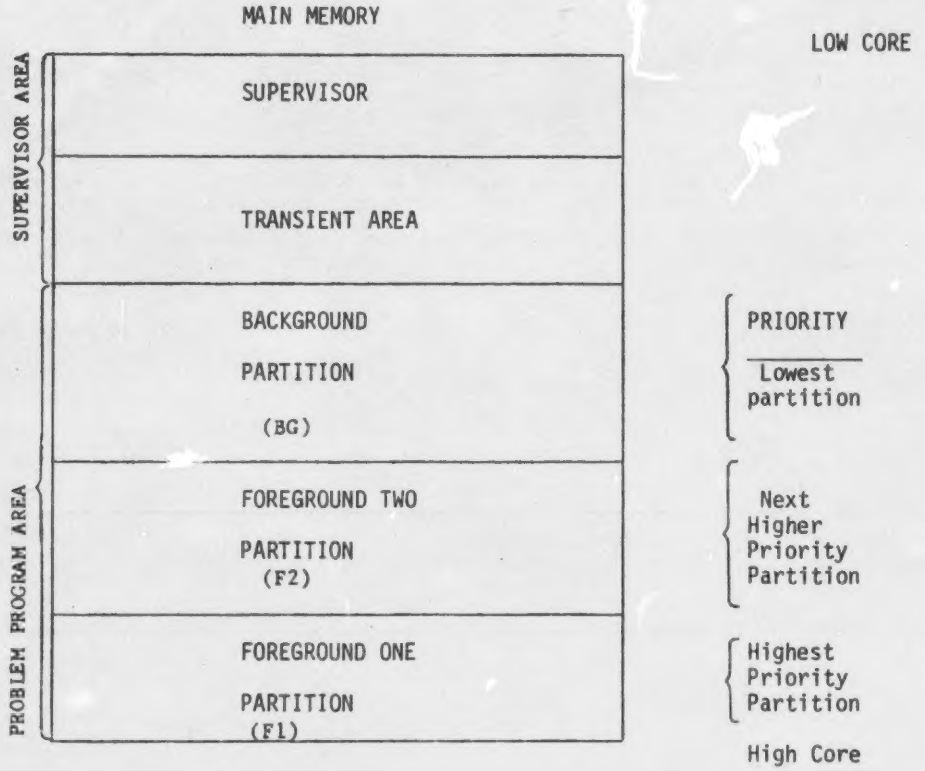
The Job Control Program is written over in the partitions by the problem program being loadedinto the partition by the System Loader. When the problem program ends execution (Stop Run), control is given to the Supervisor. The supervisor reloads the Job Control Program into the partition of the program that ended execution wiping out the program. Job Control then reads SYSRDR to obtain the Job Control statements for the next task to be performed. This process continues until all jobs in the input have been processed through the computer.

484

JOB LOADING AND EXECUTING


MAIN MEMORY
000000
SUPERVISOR NUCLEUS
SYSLDR
Physical Transient Area
Logical Transient Area
JOB
CONTROL
BG
F2
F1
SYSRES
IPL
VOL
Directory
CORE IMAGE LIBRARY
Directory
RELOCATABLE LIBRARY
Directory
SOURCE STATEMENT LIBRARY

MULTIPROGRAMMING - concurrent excution of two or more programs simultaneously residing in core storage of a computer.


MAIN MEMORY
LOW CORE
SUPERVISOR AREA
SUPERVISOR
TRANSIENT AREA
PROBLEM PROGRAM AREA
BACKGROUND
PARTITION
(BG)
PRIORITY
Lowest partition
FOREGROUND TWO
PARTITION
(F2)
Next Higher Priority Partition
FOREGROUND ONE
PARTITION
(F1)
Highest Priority Partition
High Core

486

b. Control Program is the backbone of the operating system and is composed of the following components:

(1) Job Management - identifies the job to be done and assigns I/O devices for the problem program.

(2) Task Management (supervisor) brings in the problem program from the library and responds to any error conditions by unloading one program and going on to the next one.

c. Data Management - handles movement of data between main storage and I/O devices. This also includes:

(1) Scheduling and programing the operation of channels.

(2) The identifications of data by volume (the device in which data is stored such as a reel of tape, or a disk pack)

(3) The resolution of error conditions that occur during the movement of data.

(4) Data Management is composed of the Input Output Control System, (referred to as IOCS)

(a) IOCS is a set of routines (programs) that handle the movement of data between main storage and I/O devices. These IOCS routines fall into two categories:

(1) Physical IOCS (PIOCS) - is composed of those I/O routines which supervise the reading and writing of data on I/O devices without regard for its logical content, format or organization.

(II) Logical IOCS (LIOCS) - Once the data has been brought from the I/O device by PIOCS, the LIOCS routines now operate on this data to make its logical content available to the problem program as required.

(b.) File Access Methods.

(I) Sequential access method - sequential processing is used to read/write and process successive records in a logical file.

(II) Index sequential access method - an index sequential file is built from sequential input and has indexes that permit individual records to be found for subsequent processing operations. By supplying the key of any record, a programmer can obtain the specific record randomly.

(III) Direct access method - records are usually organized in a random manner and are processed by referring to a record location reference supplied in the problem program.

This location reference indicates the exact record by track and key or identifier.

(IV) Basic Teleprocessing Access Method (BTAM) controls transmission and reception of messages over the communication lines in response to READ and WRITE macros. The LIOCS technique of accessing the file is not extended to the problem program.

(V) Queued Teleprocessing Access Method - has the same capabilities as BTAM with the addition of LIOCS technique of accessing the file by the problem program.

(c) Processing Programs.

(I) Language processor - are programs which use as input source programs and interpret them into a machine compatible language.

- (a) FORTRAN compiler
- (b) RPG compiler
- (c) COBOL compiler
- (d) PL/I compiler
- (e) ALC assembler
- (f) GPSS compiler

(d) Utility programs - generalized programs which, through the use of a control card, can perform a specific function.

(I) Card to tape utility

(II) Tape to print utility

(III) Disk to print utility

(VI) Sort/Merge ...etc.

(V) Autotest.

(e) Service programs - generate the system, create and maintain the library sections, and edit programs into disk residence before execution.

(I) Link Editor Program - prepares programs for entry into the core image library by:

(a) Assigning final main storage addresses to data and instructions.

(b) Incorporating subroutines from the relocatable library, as requested by the programer and combining separately written sections of the program into a single unified program.

(II) Librarian -Is a group of programs that maintains and reorganizes the disk library areas and provides printed and punched output from the libraries. Three libraries are used.

(a) Core Image Library. All programs in the system (IBM - supplied and user programs) are in machine code and loaded from this library by the system loader routine of the supervisor.

(b) Relocatable Library: This library stores object modules that can be used for subsequent linkage with other programs modules. A complete program of one or more modules can be placed in this library.

(c) Source Statement Library. This library stores IBM-supplied MACRO definitions and user defined source statement routines (such as MACRO definitions) built to provide extended program-assembly capability.

(III) Librarian Programs;

MAINT

CSERV

SSERV

RSERV

CORGZ

LISTVTOC

DSERV

DOS - MAIN STORAGE ORGANIZATION

SUPERVISOR Minimum 6144 Bytes	PERMANENT ROUTINES	PERMANENT STORAGE - LOCATIONS PSW'S, CSW, CAW, ETC	
		I/O UNIT, CONTROL TABLES LUB, PUB, JIB	
		COMMUNICATIONS REGION - 46 BYTES	
		CHANNEL SCHEDULER	EXCP ROUTING I/O INTERRUPT RTE START I/O RTE
		OTHER INTERRUPT ROUTINES	SVC PROGRAM CHECK MACHINE CHECK EXTERNAL
		SYSTEM LOADER (SYSLDR)	
		RESIDENT (2311) ERROR RECOVERY ROUTINES	
		STORAGE PROTECTION (OPTIONAL)	
		TIMER SERVICES (OPTIONAL)	
	TRANSIENT ROUTINES	OPEN, CLOSE, DUMP, CHECKPOINT, OPERATOR COMMUNICATIONS, ERROR PROCESSING RTES	
PROBLEM PROGRAM AREA (MINIMUM 10K BYTES)		USER PROGRAM or JOB CONTROL or LINKAGE EDITOR or LIBRARIAN ROUTINE or LANGUAGE TRANSLATOR	

0 7	DATE
8 9	ADDRESS: PROBLEM PROGRAM LABEL AREA
10 11	END OF SUPVR STORAGE, PROTECTION
12 22	USER AREA
23	PROGRAM SWITCHES (UPSI)
24 31	JOB NAME
32 35	ADDRESS: UPPERMOST BYTE OF PROBLEM PROGRAM AREA
36 39	ADDRESS: UPPERMOST BYTE OF PROBLEM PROGRAM PHASE
40 43	ADDRESS: UPPERMOST BYTE OF LONGEST PROBLEM PROGRAM PHASE
44 45	LENGTH OF PROBLEM PROGRAM LABEL AREA

The Communication Region

Diagram of PHYSICAL UNIT BLOCK (PUB) - Where the machine address of the I/O device can be found:

CHANNEL	BYTE 0
UNIT	BYTE 1
POINTER TO CHANNEL QUEUE	BYTE 2
ERROR RETRY COUNTER OR POINTER TO TAPE ERROR BLOCK (BOTH TEB AND TEBV)	BYTE 3
DEVICE TYPE	BYTE 4
DEVICE OPTIONS (TAPE SET MODE, ETC)	BYTE 5
CHANNEL SCHEDULER FLAGS	BYTE 6
JOB CONTROL FLAGS	BYTE 8

LOGICAL UNIT BLOCK (LUB) - Where system and programmer logical units are defined:

Card Reader (JCL)	*SYSRDR
Card Reader (DATA)	*SYSIPT
Printer	*SYSLST
Console Type Writer	*SYSLOG
Linkage Editor Work Area	SYSLNK
Disk Containing the Operating System	SYSRES
Private Source Statement Library	SYSSLB
Private Relocatable Library	SYSRLB
Private Core Image Library	SYSCLB
Card Punch	*SYSPCH
Programmer Logical Units	SYS000 - SYSMAX

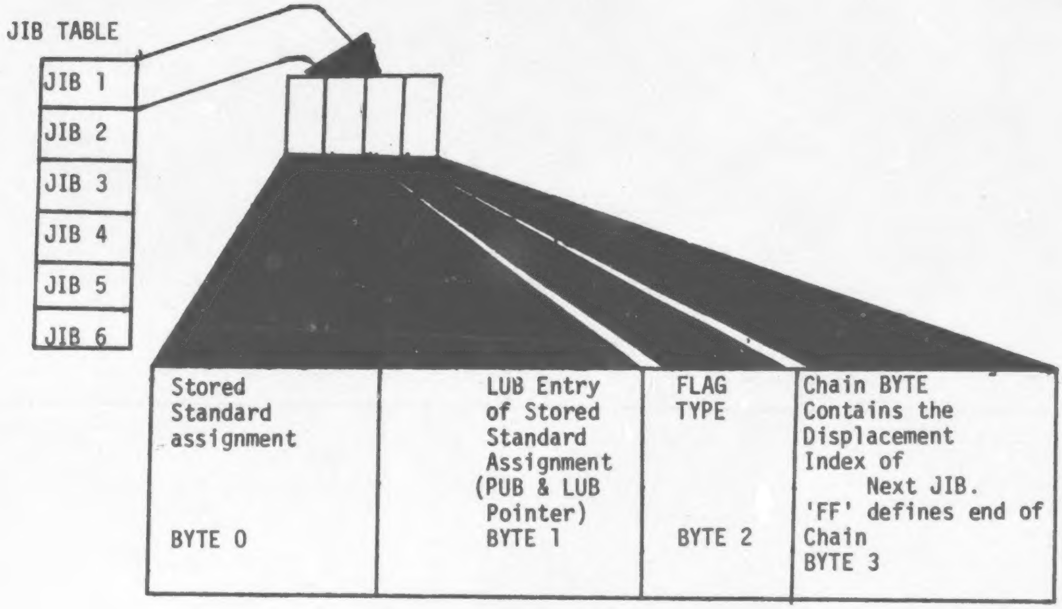
BYTE 0	BYTE 1
Pointer to Pub	Pointer to Job Information Block

* May be used in DTF of ALC

JOB INFORMATION BLOCK (JIB)

The JIB contains one of the following:

1. LUB entry of the standard assignment when a temporary LUB assignment is made.
2. PUB pointer for an alternate LUB assignment.
3. Extent information when DASD files protection is selected as a supervisor generation option.



Stored Standard assignment BYTE 0	LUB Entry of Stored Standard Assignment (PUB & LUB Pointer) BYTE 1	FLAG TYPE BYTE 2	Chain BYTE Contains the Displacement Index of Next JIB. 'FF' defines end of Chain BYTE 3

494

JOB CONTROL LANGUAGE (JCL)

Whereas COBOL is a language for man-machine communication, JCL is a language for man-operating system communication.

JCL can enter the system through many channels. JCL cards may be keypunched by the programmer and placed into the system through the card reader along with such items as a COBOL source deck, a punched object deck, data cards or any combination of these. JCL may be cataloged in the Source Statement library to be invoked by SPOOLing programs. JCL may be entered into the system by the operator thru the computer console device.

STAGES OF PROGRAM DEVELOPMENT

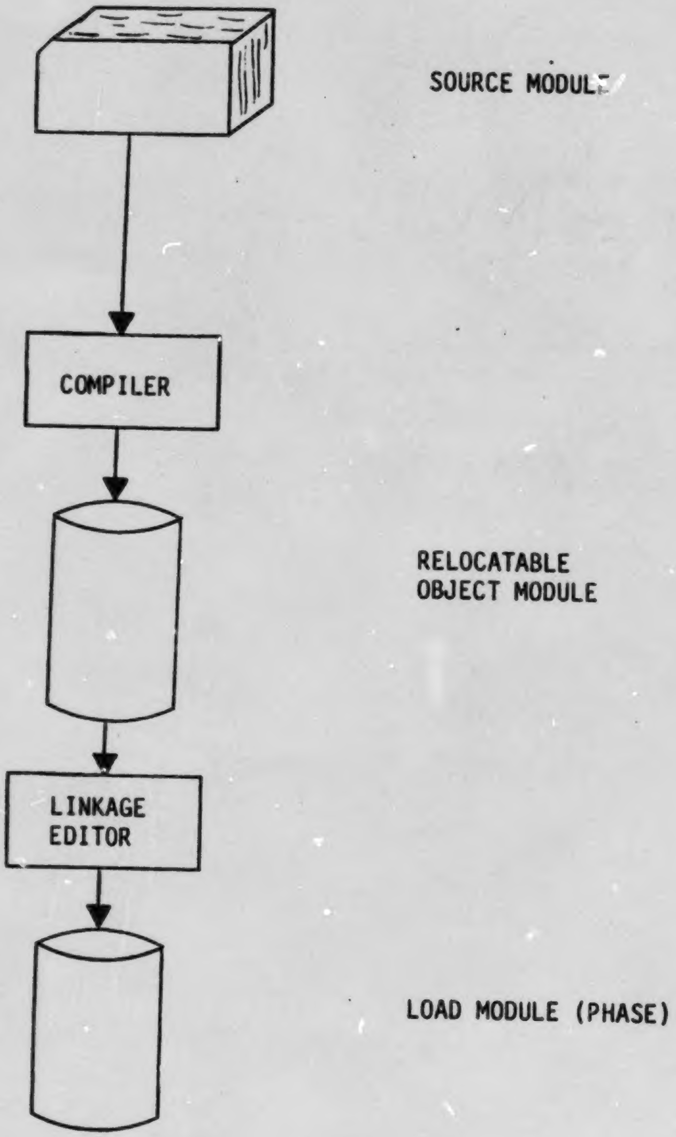
The term program is used to describe several things. The programmer codes sets of source statements that may be a complete program or part of a program. These source statements are then compiled into a relocatable machine-language program which, in turn, must be edited into an executable program, and may be combined with other programs. Consequently, it is convenient to refer to each stage of program development by a particular name.

A set of source statements that is processed by a language translator (Assembler, COBOL, FORTRAN, RPG or PL/I), is referred to as a source module or source deck (if in card form).

The output of a language translator is referred to as an object module (or deck). All object modules must be further processed by the linkage editor before they can be executed in the system.

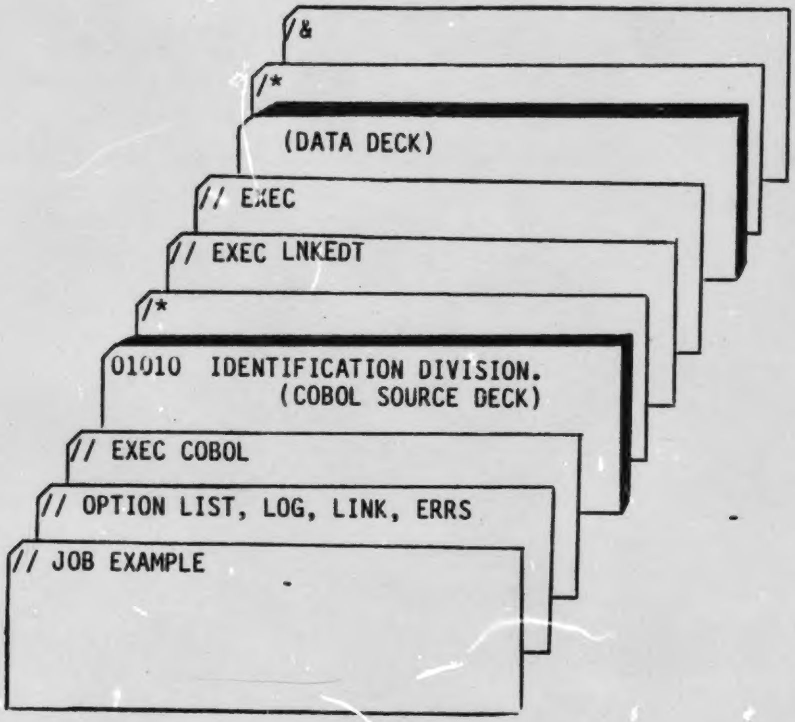
The output of the linkage editor consists of one or more program phases in the core-image library. A phase is in executable, nonrelocatable, core image form.

THREE PROGRAM FORMS


SOURCE MODULE
COMPILER
RELOCATABLE OBJECT MODULE
LINKAGE EDITOR
LOAD MODULE (PHASE)

COMPILE - LINKEDIT - EXECUTE

The following represents the JCL for a very simple program that reads cards, does some sort of calculations, and prints out a report. The cards bounded by the //JOB and the /& represent what is called a job. This particular job consists of 3 job steps each of which has a //EXEC card associated with it.



```
/&
/*
     (DATA DECK)
// EXEC
// EXEC LNKEDT
/*
01010  IDENTIFICATION DIVISION.
              (COBOL SOURCE DECK)
// EXEC COBOL
// OPTION LIST, LOG, LINK, ERRS
// JOB EXAMPLE
```

// JOB EXAMPLE

The JOB statement indicates the beginning of control information for a job.

FORMAT

```
// JOB     jobname   [accounting information]
```

jobname	The name of the job. Must be 1 to 8 alphameric characters the first of which must be alphabetic.
accounting information	If the job accounting interface has been specified during system generation, 16 characters of user specified accounting information can be entered in the job statement. It must be separated from the jobname by a blank.

// OPTION LIST,LOG,LINK,ERRS

The OPTION statement specifies one or more of the Job Control options.

FORMAT

```
// OPTION  option 1, option 2,.....
```

The purpose of this card is to overide standard system options set at generation time. Selected options can be in any order. Options are reset to the system standards at the end of each job. (INDICATED BY & CARD)

LOG	List all JCL on the printer from this card until a NOLOG, // JOB, or /& is encountered.
NOLOG	Stop listing JCL.
DUMP	Causes a dump of registers and main storage to be printed on printer if the program abnormally terminates.
NODUMP	Turn off the dump option.

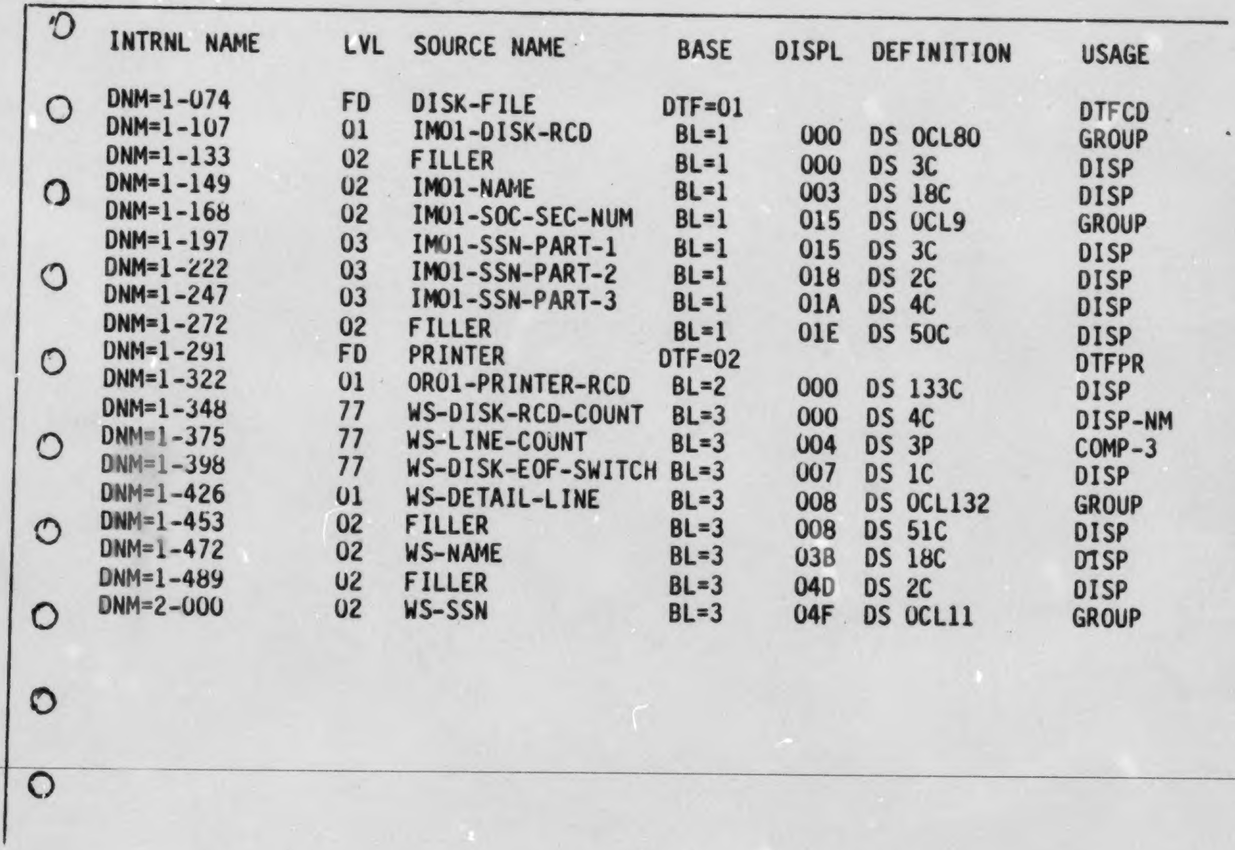
LINK	Tells the COBOL compiler to put the object module in the linkage editor work area known as SYSLNK.
NOLINK	Turn off link option.
DECK	Tells the COBOL compiler to punch an object deck.
NODECK	Turn off the deck option.
LIST	Tells the COBOL compiler to write the source listing on PRINTER
NOLIST	Turn off the list option.

```
    05  FILLER                                   PIC X(82)
                                         VALUE IS
        ' IS THE TOTAL NUMBER OF RECORDS  PRINTED'.
01  WS-PAGE-HEADING.
    05  FILLER                                   PIC X(56)
                                         VALUE IS SPACES.
    05  FILLER                                   PIC X(7)
                                         VALUE IS 'N A M E'.
    05  FILLER                                   PIC X(8)
                                         VALUE IS SPACES.
    05  FILLER                                   PIC X(62)
                                         VALUE IS 'SOC-SEC-NUM'.
PROCEDURE DIVISION
    OPEN INPUT  DISK-FILE
         OUTPUT PRINTER.
    PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT
      UNTIL WS-DISK-EOF-SWITCH EQUAL TO '1'.
    MOVE WS-DISK-RCD-COUNT TO WS-TL-DISK-RCD-COUNT.
    WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
      AFTER ADVANCING 3 LINES.
    CLOSE DISK-FILE
         PRINTER.
    STOP RUN.
0010-READ-AND-PRINT.
    READ DISK-FILE
      AT END
        MOVE '1' TO WS-DISK-EOF-SWITCH
```

LISTX	Tells the COBOL compiler to write a PROCEDURE DIVISION MAP on SYSLST.
NOLISTX	Turn off listx option.

```
 86   000726                        START   EQU   *
      000726  41 10 C 045                   LA    1,045(0,12)          LIT+13
      00072A  58 00 D 1C8                   L     0,1C8(0,13)          DTF=1
      00072E  18 40                         LR    4,0
      000730  07 00                         BCR   0,0
      000732  05 F0                         BALR  15,0
      000734  50 00 F 008                   ST    0,008(0,15)
      000738  45 00 F 00C                   BAL   0,00C(0,15)
      00073C  00000000                      DC    X'00000000'
      000740  0A 02                         SVC   2
      000742  41 10 C 045                   LA    1,045(0,12)          LIT+13
      000746  58 00 D 1CC                   L     0,1CC(0,13)          DTF=2
      00074A  18 40                         LR    4,0
      00074C  07 00                         BCR   0,0
      00074E  05 F0                         BALR  15,0
      000750  50 00 F 008                   ST    0,008(0,15)
      000754  45 00 F 00C                   BAL   0,00C(0,15)
      000758  00000000                      DC    X'00000000'
      00075C  0A 02                         SVC   2
      00075E  50 20 D 1C0                   ST    2,1C0(0,13)          BL=2
      000762  58 80 D 1C0                   L     8,1C0(0,13)          BL=2
 88   000766  58 00 D 1E8                   L     0,1E8(0,13)          VN=01
      00076A  50 00 D 1E4                   ST    0,1E4(0,13)          PSV=1
      00076E  58 00 C 010                   L     0,010(0,12)          GN=01
      000772  50 00 D 1E8                   ST    0,1E8(0,13)          VN=01
      000776                        GN=01   EQU   *
      000776  58 20 C 014                   L     2,014(0,12)          GN=02
      00077A  D5 00 6 007 C 04D             CLC   007(1,6),04D(12)     DNM=1-398
      000780  07 82                         BCR   8,2
      000782  58 10 C 004                   L     1,004(0,12)          PN=01
      000786  07 F1                         BCR   15,1
      000788                        GN=02   EQU   *
      000788  58 00 D 1E4                   L     0,1E4(0,13)          PSV=1
```

SYM	Tells the COBOL compiler to write a DATA DIVISION MAP on the printer.
NOSYM	Turn off SYM option.



```
INTRNL NAME    LVL  SOURCE NAME          BASE    DISPL  DEFINITION   USAGE

DNM=1-074      FD   DISK-FILE            DTF=01                      DTFCD
DNM=1-107      01   IM01-DISK-RCD        BL=1    000    DS OCL80     GROUP
DNM=1-133      02   FILLER               BL=1    000    DS 3C        DISP
DNM=1-149      02   IM01-NAME            BL=1    003    DS 18C       DISP
DNM=1-168      02   IM01-SOC-SEC-NUM     BL=1    015    DS OCL9      GROUP
DNM=1-197      03   IM01-SSN-PART-1      BL=1    015    DS 3C        DISP
DNM=1-222      03   IM01-SSN-PART-2      BL=1    018    DS 2C        DISP
DNM=1-247      03   IM01-SSN-PART-3      BL=1    01A    DS 4C        DISP
DNM=1-272      02   FILLER               BL=1    01E    DS 50C       DISP
DNM=1-291      FD   PRINTER              DTF=02                      DTFPR
DNM=1-322      01   OR01-PRINTER-RCD     BL=2    000    DS 133C      DISP
DNM=1-348      77   WS-DISK-RCD-COUNT    BL=3    000    DS 4C        DISP-NM
DNM=1-375      77   WS-LINE-COUNT        BL=3    004    DS 3P        COMP-3
DNM=1-398      77   WS-DISK-EOF-SWITCH   BL=3    007    DS 1C        DISP
DNM=1-426      01   WS-DETAIL-LINE       BL=3    008    DS OCL132    GROUP
DNM=1-453      02   FILLER               BL=3    008    DS 51C       DISP
DNM=1-472      02   WS-NAME              BL=3    03B    DS 18C       DISP
DNM=1-489      02   FILLER               BL=3    04D    DS 2C        DISP
DNM=2-000      02   WS-SSN               BL=3    04F    DS OCL11     GROUP
```

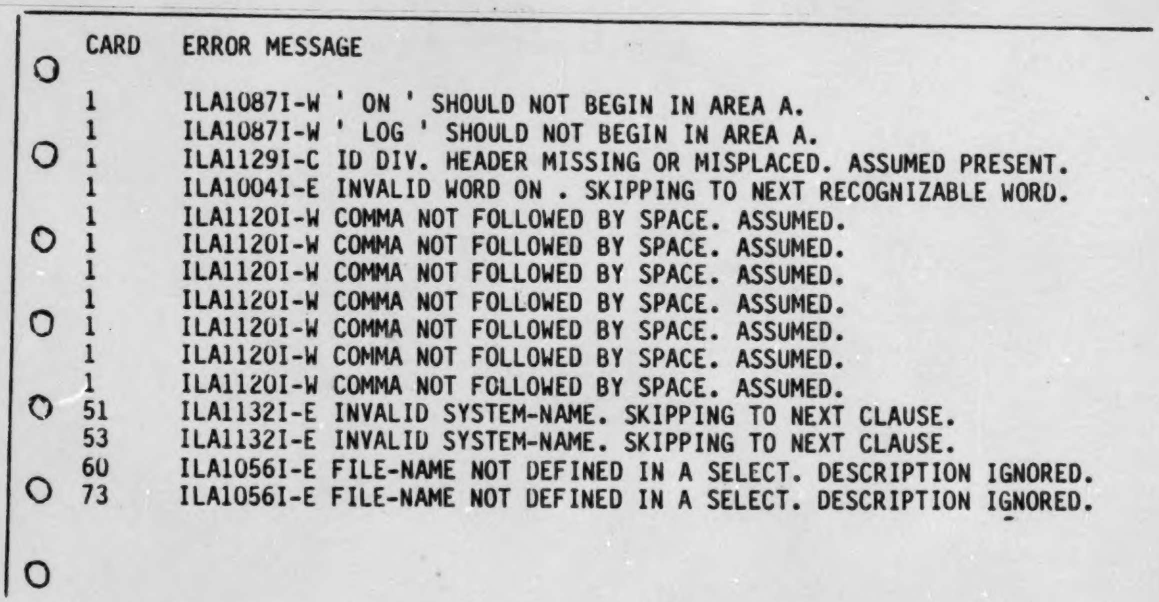
XREF Tells the COBOL compiler to write the symbolic cross reference list on the printer.

NOXREF Turn off XREF option

```
                              CROSS-REFERENCE DICTIONARY

DATA NAMES             DEFN    REFERENCE

DISK-FILE              00025   00086       00086   00093   00097   00097
IM01-NAME              00038   00109
IM01-SSN-PART-1        00040   00106
IM01-SSN-PART-2        00041   00107
IM01-SSN-PART-3        00042   00108
PRINTER                00027   00086       00086   00091   00103   00110
OR01-PRINTER-RCD       00046   00091       00091   00103   00103   00103   00110   00110
WS-DISK-RSD-COUNT      00048   00090       00101   00101
WS-LINE-COUNT          00049   00102       00105   00112   00112
WS-DISK-EOF-SWITCH     00051   00088       00099
WS-DETAIL-LINE         00053   00110       00110
WS-NAME                00056   00109
WS-SSN-PART-1          00060   00106
WS-SSN-PART-2          00063   00107
WS-SSN-PART-3          00066   00108
WS-TOTAL-LINE          00069   00091       00091
WS-TL-DISK-RCD-COUNT   00072   00090
WS-PAGE-HEADING        00076   00103       00103

PROCEDURE NAMES        DEFN    REFERENCE

0010-READ-AND-PRINT    00096   00088
0010-EXIT              00113   00088       00100
```

ERRS Tell the COBOL compiler to summarize all errors in the source program on the printer.

NOERRS Turn off the ERRS option.



```
CARD   ERROR MESSAGE

1      ILA1087I-W ' ON ' SHOULD NOT BEGIN IN AREA A.
1      ILA1087I-W ' LOG ' SHOULD NOT BEGIN IN AREA A.
1      ILA1129I-C ID DIV. HEADER MISSING OR MISPLACED. ASSUMED PRESENT.
1      ILA1004I-E INVALID WORD ON . SKIPPING TO NEXT RECOGNIZABLE WORD.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1      ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
51     ILA1132I-E INVALID SYSTEM-NAME. SKIPPING TO NEXT CLAUSE.
53     ILA1132I-E INVALID SYSTEM-NAME. SKIPPING TO NEXT CLAUSE.
60     ILA1056I-E FILE-NAME NOT DEFINED IN A SELECT. DESCRIPTION IGNORED.
73     ILA1056I-E FILE-NAME NOT DEFINED IN A SELECT. DESCRIPTION IGNORED.
```

CATAL Causes the program being link edited to be cataloged into the core-image library.

The Standard System Options of Student jobs are as follows: NOLOG, NODUMP, NOLINK, NODECK, LIST, NOLISTX, NOSYM, ERRS, XREF.

504

```
// EXEC FCOBOL
```

The EXEC statement indicates the end of control information for a job step and that execution of a program is to begin.

FORMAT

```
// EXEC   progname
```

Progname   Is the name of the program cataloged in the core image library.

What occurs here is that the progname is passed to the supervisor which goes to the core-image library directory, finds the program, loads it into memory, and passes control to it to begin executing.

NOTE: FCOBOL is the name of a COBOL compiler.

```
source deck
```

This is the COBOL source deck that is to be compiled.

```
/ *
```

This delimiter card must be the last statement of each input data file. It causes the system to return an end-of-file indication to the program utilizing the data file.

```
// EXEC LNKEDT
```

This execute statement causes the linkage editor to be loaded into memory and control passed to it. The linkage editor looks for its input on SYSLNK. The output from the linkage editor will be placed in the temporary portion of the core-image library unless the OPTION CATAL is specified.

```
// EXEC
```

This execute statement with no name in the operand field causes the program just placed in the temporary portion of the core-image library to be loaded into memory and executed.

```
Data deck
```

These are the data cards that are to be processed by the program that was just compiled and link edited.

```
/*
```

End of data cards.

```
/&
```

End of job - all options reset to system standards.

EXERCISES

1. Write the JCL to compile a COBOL program and produce only an object deck, a source listing, an error listing, and a cross-reference listing. Assume all options are set in the NO_ _ _ mode for all exercises in the course.

2. Write the JCL for a single job to compile two COBOL programs. The first should produce only an object deck and the second should produce only a source listing, an error listing, a cross-reference listing and a data division map.

3. Write the JCL to compile and execute a COBOL program. Produce a source listing, an error listing, a cross-reference listing and if the program abnormally terminates, a core dump should be provided.

THE LINKAGE EDITOR

The linkage editor combines object modules supplied in the input job stream, and/or newly compiled object modules on SYSLNK, and/or object modules from a relocatable library. It edits these modules into executable programs.

The linkage editor puts these executable programs into one of two places. If the option CATAL has been specified the program is cataloged as a member of a core-image library. If not the program is placed in the temporary portion of the core-image library.

The compile-linkedit-execute example showed the linkage editor taking a just compiled program and linking it into the temporary portion of the core-image library.

If a program phase is to be cataloged into the core-image library a linkage editor control card is required. The card is a 'PHASE' card and the format is as follows.

```
ƀ PHASE    name,origin
```

ƀ	Card column 1 must be blank.
PHASE	The operation code
name	1 to 8 alphameric characters used to give a name to the phase being cataloged. When the program phase is executed at a later date this is the name to be used in the execute card. (// EXEC name
,origin	This operand allows the user to specify the load address to be used for this program phase. The possible options are; 1. symbol [(phase)][±relocation] 2. * [±relocation] 3. S [±relocation] 4. ROOT 5. + displacement 6. F + address 1. Used when building a program overlay structure. Says to link edit this phase at the same address as a previously linked phase with optional relocation shift.

2. Used when more than one phase is being linked together. Says put this one right behind the one just link edited. In a single phase operation this is the same as 'S'.

3. Says link this phase to load at the beginning of the partition we are working in.

4. Used in overlay and says that this particular phase is not to be overlayed but is to remain in memory while program is executing.

5. Specifies an absolute address.

6. Used to link edit a program to run in a foreground partition while the linkage editor is executing in background.

If object modules from the relocatable library or object decks external to the system are to be link edited, another control card comes into play.

b̸ INCLUDE [modulename]

modulename Symbol name of the module as used when cataloged in the relocatable library. If the operand is omitted the system assumes that the module is in the input job stream immediately behind the INCLUDE card.

The third linkage editor control card is the ENTRY card and is in the following format:

b̸ ENTRY entrypoint

entrypoint Symbolic name of the address where control is to be passed when the program is loaded into memory.

Note: The ENTRY card is normally not needed. Job control will write an ENTRY statement when EXEC LNKEDT is read to ensure that an ENTRY statement will be present to halt linkage editing.

The last of the linkage editor control cards is ACTION card and is in the following format:

```
ƀ ACTION        CLEAR,MAP,NOMAP,BG,F1,F2
```

CLEAR	Indicate that the unused portion of the core image library will be set to Binary zero before the beginning of the linkage editor function. CLEAR is very time consuming and should not be used. If you are careful to assign initial values where needed in your Program this isn't needed.
MAP	Indicates that PRINTER is available for diagnostic messages. In addition, a main storage map it output on PRINTER.

```
24/06/77   PHASE   XFR-AD LOCORE  HICORE  DSK-AD  ESD TYPE  LABEL     LOADED  REL-FR

        PHASE*** 005000 005000  005AF0  30 13 3 CSECT     EXAMPLE0  005000  005000

                                                 CSECT     IJCFZII0  005950  005950

                                                 CSECT     IJDFAPIZ  0059D8  0059D8
                                                 * ENTRY   IJDFAZIZ  0059D8

                                                 CSECT     ILBDMNS0  005AF0  005AF0
```

NOMAP	Diagnostics are listed on SYSLOG.
BG F1 F2	Tells the linkage editor what Partition the program is being link edited to run in.

EXERCISES

1. You are handed a deck of cards and told that is is an object module received from Fort Knox DPI. Write the JCL necessary to execute the program.

2. Write the JCL needed to catalog the object module mentioned in exercise 1 into the core-image library. Call the program FKX512.

3. Write the JCL to execute the program cataloged in exercise 2.

ASSIGNMENT OF I/O DEVICES

In order for a program to run in a computer system some means must be available to LINK the program to the physical devices such as card readers, printers, and disk drives.

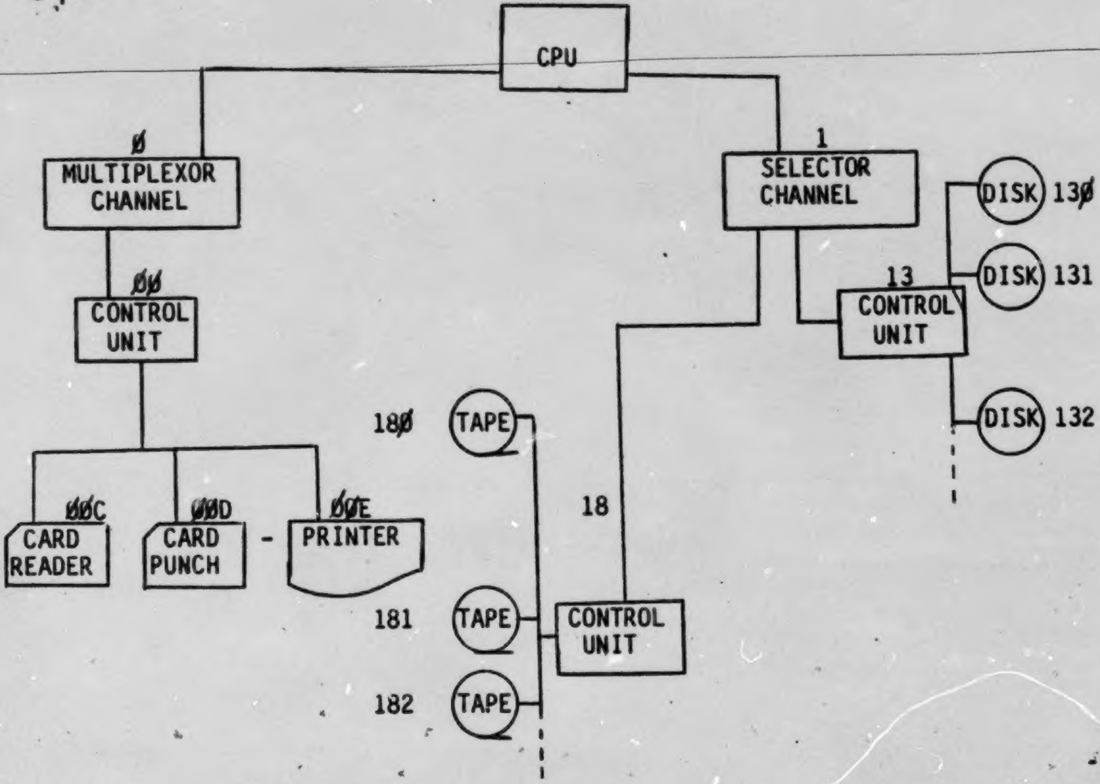
The programmer is given the latitude to utilize any symbolic device address he wishes, ranging from SYS000 through SYSmax. For example in a COBOL program you may choose to call the printer 'SYS005' and the card reader 'SYS006' and the disk unit you are using 'SYS010'. The preceding is accomplished through the use of the COBOL SELECT statement.

```
FILE-CONTROL
    SELECT PRINT-FILE
        ASSIGN TO SYS005-UR-1403-S.
```

Because this program could be required to run on many different hardware configurations the physical device addresses are likely to be different. The following diagram depicts a typical hardware configuration indicating the physical device addresses.

NOTE: The // ASSGN card discussed later in this section serves the function of linking the program logical assignments to the hardware configuration.

512


CPU
0
MULTIPLEXOR CHANNEL
00
CONTROL UNIT
00C
CARD READER
00D
CARD PUNCH
00E
PRINTER
1
SELECTOR CHANNEL
13
CONTROL UNIT
DISK 130
DISK 131
DISK 132
180
TAPE
181
TAPE
182
TAPE
18
CONTROL UNIT

There exists in the DOS system a set of fixed symbolic names used to reference various I/O devices. These are:

SYSRDR -	Card reader, magnetic tape unit, or disk extent (specific area) used for JOB control input.
SYSIPT -	Card reader, magnetic tape or disk extent used as input for programs.
SYSPCH -	Card punch, magnetic tape or disk extent used for punched output.
SYSLST -	Printer, magnetic tape or disk extent used for punched output.
SYSLOG -	Printer-keyboard or console printer-keyboard used for operator messages and to log JCL message.

SYSLNK - Disk extent used as input to the linkage editor.

SYSRES - System residence area on a disk drive.

SYSCLB - Disk extent used for a private core image library.

SYSRLB - Disk extent used for a private relocatable library.

SYSSLB - Disk extent used for a private source statement library.

SYSREC - Disk extent used to store error records collected by the system.

SYS000 - SYSmax - All other units on the system.

The first eleven of these symbolic names, called system logical units, are used by the system control programs and system service programs.

The remainder SYS000, SYS001, ..., SYSmax are known as programmer logical units. Some of these are defined and assigned to physical devices at system generation time such as the work areas required by the COBOL compiler.

The following are the physical device addresses for the ADMINCEN computer system and some of the Standard logical assignments set at system generation time:

PHYSICAL ADDRESS	DEVICE	LOGICAL ASSIGNMENT
00C	CARD READER	SYSRDR, SYSIPT, SYS020
00D	CARD PUNCH	SYSPCH, SYS021
00E	PRINTER	SYSLST, SYS022
01F	CONSOLE TYPEWRITER	SYSLOG
180-185	6 TAPE DRIVES	NONE
130	DISK DRIVE	SYSRES, SYSREC
131	DISK DRIVE	SYSLNK, SYS000 thru SYS009
132	DISK DRIVE	SYSSLB, SYSCLB
133-136	DISK DRIVE	NONE

Each of the disk assignments has associated with it information telling the system where exactly on the disk pack the specified file resides. For example Disk drive 131 has eleven different areas on the pack assigned to SYSLNK and SYS000 thru SYS009, they all represent different areas some of which are used by the COBOL compiler for work areas.

514

```
// ASSGN  SYS005,X'00C'
```

The ASSGN statement assigns a logical I/O unit to a physical device. It remains in effect until the next change in assignment or until the end of job.

```
// ASSGN  SYSxxx,address    {,x'ss'}
                            { ,ALT }
```

SYSxxx	The programmer logical unit used in your program.
address	The physical device address in the form of X'cuu' which indicates the Channel and Unit number (in hexadecimal). CHANNEL (c) 0 for multiplexor channel, 1-6 for selector channels 1 thru 6. UNIT (uu) 00 - FE (0 to 254) in Hexidecimal
x'ss'	used to tell the system the physical characteristics of a tape unit such as bytes per inch and parity used. Normally this information is provided at system generation time and is not needed unless changes have been made.
ALT	Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The specifications are the same for primary and alternate tape drives.

```
// ASSGN SYS010,X'180'
// ASSGN SYS010,X'181', ALT
```

// ASSGN cards must be placed in the job stream prior to the // EXEC that utilizes them and should be placed after the previous // EXEC in the job stream. The reason for the latter is to group all of the associated JCL with the Job step. In the case of multistep jobs JCL changes are difficult if all of the 'ASSGN' cards are bunched together at the beginning of the first step.

AUTOMATIC DEVICE ASSIGNMENT METHOD (ADAM)

The DOS ASSGN that we have just discussed does have some limitations. One of the limitations is that the JCL, which is written ahead of time,is hardware system dependent and would have to be modified card-by-card if the same set of JCL were to be used on a different hardware configuration. This does not present a great problem for programs that are written and run on the same machine. But, it does present a problem for Army Standard systems that are witten in a central location along with the associated JCL and sent out to the world to be run on many different machines. A second limitation is the ability to get around an I/O device malfunction. It must be done in the same manner as above, a card-by-card change in the JCL ASSGN's.

To overcome these limitations ADAM was written to provide the ability to make DOS JCL hardware configuration independent and return system resources to the DOS console operator for management.

EXAMPLE:

```
// ASSGN SYS010,X'TP0'
```

TP0 Positions 1 and 2 indicate the particular device type;

CR	card reader
CP	card punch
PR	printer
DK	disk drive
TP	tape drive
TD	tape drive w/dual density feature
T7	tape drive seven track

The 3rd position indicates the positional relationship of this device to any other device of the same classification. For example; TP0,TP1,TP2,TP3,..... or CR0,CR1,PR0,PR1......

There is a SYMBOLIC UNIT TABLE in memory that relates these symbolic assignments to the actual device addresses. The table logically appears as follows;

SYMBOL	BG	F2	F1
CR0	00C	00C	00C
CP0	00D	00D	00D
PR0	00E	00E	00E
TP0	180	182	184
TP1	181	183	185
TP2	182	184	180
TP3	183	185	181
TP4	184	180	182
TP5	185	181	183
DK1	130	134	136
.	.	.	.
.	.	.	.

In the example above we saw that the program called for 'TP0'. As you can see from the table if the program were run in the background (BG) partition it would be assigned to tape drive '180'. If it were run in foreground 2 (F2) it would be assigned to tape drive '182'. On some other hardware configuration it may be an address that does not even appear in this table.

516

If device 180 were to become inoperative for some reason, the operator would change the assignment in the symbolic unit table with the following three commands entered thru the console device;

```
SYMCHG  TP0,BG=185
SYMCHG  TP4,F2=183
SYMCHG  TP2,F1=181
```

This is certainly easier than hunting up all of the `// ASSGN SYSxxx,X'180'` cards and repunching them.

The Symbolic Unit Table takes about 30 seconds to build at IPL time.

EXERCISES

1. Write the JCL to compile and execute a COBOL program which uses the following SELECT statements

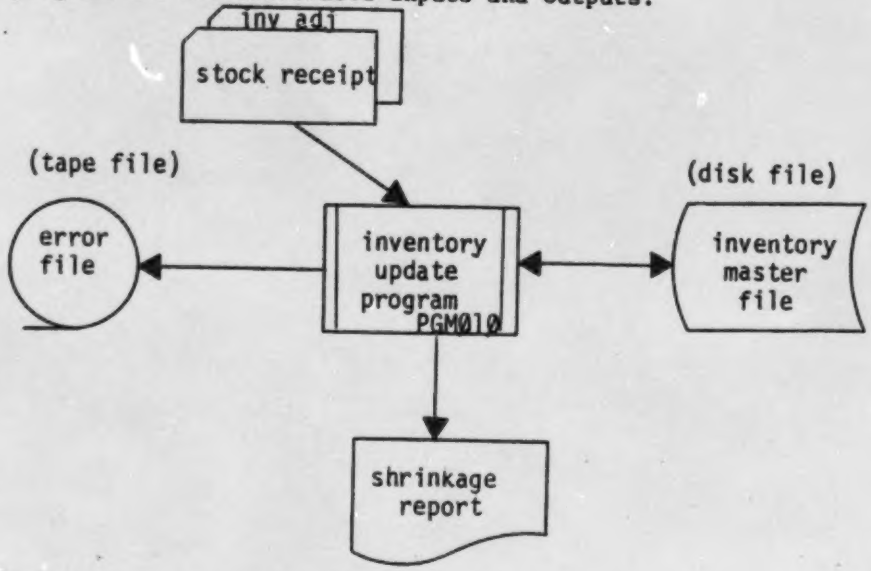
```
SELECT CARD-FILE ASSIGN SYS008-UR-2540R-S.
SELECT PRINT-FILE ASSIGN SYS009-UR-1403-S.
```

2. Write the JCL to execute an <u>object</u> deck for a program which reads cards and produces a magnetic tape file and a printed report. The programmers logical units used by the program are SYS005 for the card reader, SYS008 for the printer, and SYS010 for the tape file.

3. Write the JCL for two COBOL programs which are to be compiled and executed as separate job steps within the same job. The first program while executing reads a card file and punches an output card file. The second program reads another set of data cards and produces a printed report. The first program uses SYS005 for the card reader and SYS007 for the card punch. The second program uses SYS006 for the card reader and SYS005 for the printer.

DATA FILES

The diagram below is a portion of a Systems Flow Chart. It depicts a single program and its associated inputs and outputs.


inv adj
stock receipt
(tape file)
error file
inventory update program
PGM010
(disk file)
inventory master file
shrinkage report

We have already seen how the // ASSGN ... cards are used to tell the operating system what devices are to be assigned to this program in order to have all of these input and output files on line. No further discussion is necessary for the CARD and PRINT files. However, because there are normally hundreds or even thousands of reels of tape and hundreds of disk packs in an organization, it is necessary to have some way of uniquely identifying each disk pack and each tape reel. The problem is compounded by the fact that there may be more than one file on each volume.

TAPE FILES

(IBM TAPES LABELS FOR DOS; ORDER NO. GC24-5070)

Tape files may appear as one or more files per volume or more than one volume per file. In any case the system must have some means of identifying the volume that it is reading and some means of identifying the file on that volume(s). This is accomplished through the use of STANDARD TAPE LABELS. Each volume has a standard volume label and a standard end-of-volume label. Each file has a standard header label and a standard end-of-file label. (See Page 6-2.1Volume Layouts)

VOLUME-LABEL

The VOL1 label is written by a utility program, Initialize Tape. It is generally written once, when the reel of tape is first received in an installation. At that time, a permanent volume serial number is assigned to the reel and written on it as part of the volume label. This provides a permanent identification of the reel, as long as it is used for files, with standard labels. (See Page 6-2.2Standard Volume Label)

FILE-HEADER-LABEL

Each tape file has a header label as the first physical record. It consists of 80 bytes of information as provided by the user and the IOCS routines to uniquely identify the file. (See Page 6-2.3Standard File Header Label)

END-OF-FILE OR END-OF-VOLUME LABEL

Either one or the other of these will appear at the end of each volume depending on the file-volume relationship. The data in these labels are the same as that in the File-Header-Label with two exceptions. Bytes 1 thru 4 will contain either EOF1 or EOV1 and bytes 55 thru 60 contain a block count which indicates the number of physical records written on it's particular volume.

Volume layouts: Standard Tape Labels

6-2.1

VOL1	HDR1	TM	DATA RECORDS	TM	EOF1	TM	TM

SINGLE-FILE-SINGLE-VOLUME

NOTE: TM is tape mark which is a special character that the tape drive writes on tapes to indicate check points.

VOL1	HDR1	TM	DATA RECORDS	TM	EOV1	TM	FIRST VOLUME

VOL1	HDR1	TM	DATA RECORDS	TM	EOV1	TM	MIDDLE VOLUME(S)

VOL1	HDR1	TM	DATA RECORDS	TM	EOF1	TM	TM	LAST VOLUME

SINGLE-FILE-MULTIVOLUME

VOL1	HDR1	TM	DATA RECORDS FILE A	TM	EOF1	TM	HDR1	TM	DATA RECORDS FILE B	TM	EOF1	TM	TM

MULTIFILE-SINGLE-VOLUME

CSU.SFW.OS

STANDARD VOLUME LABEL FOR DOS TAPE FILES.

6-2.2

VOL1	volume serial number	not used
1 4	5 10	11 80

BYTES	ENTRY
1-4	VOL1
5-10	volume serial number written when tape is initialized by Utility Program.
11-80	not used for DOS

STANDARD FILE HEADER LABEL FOR DOS TAPE FILES.

VERSION NUMBER OF GENERATION

HDR1	FILE IDENTIFIER	FILE SERIAL NUMBER	VOLUME SEQUEN NUMBER	FILE SEQUEN NUMBER	GENERA NUMBER	↓	CREAT -ION	EXPIRA -TION	NOT USED
1 4	5 21	22 27	28 31	32 35	36 39	40 41	42 47	48 53	54 80

BYTES	ENTRY	
1-4	HDR1	
5-21	File Identifier	Permits the user to identify his logical file by an application-oriented unique name.
22-27	File Serial Number	Contains the volume serial number from the volume label of the first or only volume.
28-31	Volume Sequence Number	Used for multivolume files to insure that the tape reels are mounted in the proper sequence. For Single volume files this entry defaults to 0001.
32-35	File Sequence Number	Used for multifile volumes. For single volume files this entry defaults to 0001.
36-39	Generation Number	Identifies the various editions of a file, such as grandfather-father-son relationship. Defaults to 0001 or number provided by user.
40-41	Version Number of Generation	Provides a more detailed identification of the editions of a file. Defaults to 01 or number provided by user.
42-47	Creation Date	System writes this date from IPL date entered by the operator.
48-53	Expiration Date	The date the file can be destroyed. Defaults to creation date or date as provided by user.
54-80		Not used for file header label.

```
// TLBL TAPEFLE,'TAPE MASTER',150,771834
```

The tape label card is used to provide file label information for tape label checking and writing.

FORMAT

```
// TLBL       filename,['file-id'],[date],[file-serial-number],
              [volume-sequence-number],[file-sequence-number],
              [generation-number],[version-number]
```

filename — This operand serves to link the TLBL card to the COBOL program file description (FD). It consists of 1 to 7 alphameric characters the first of which must be an alphabetic.

EXAMPLE 1:

```
SELECT TAPE-FILE1
    ASSIGN TO SYS011-UT-2400-S-TAPEFLE.
// ASSGN SYS011,X'183'
// TLBL TAPEFLE,'TAPE FILE ONE'
```

EXAMPLE 2:

```
SELECT TAPE-FILE2
    ASSIGN TO SYS012-UT-2400-S.
// ASSGN SYS012,X'184'
// TLBL SYS012,'TAPE FILE TWO'
```

'file-ID' — This optional entry consists of 1 to 17 alphameric characters, contained within apostrophes. It appears in the TLBL and the file label and serves to uniquely identify the file.

date — This optional entry enables the user to specify how long the file is to be retained and can be specified in one of three ways:

1. Omit operand in which case the file will not be retained.

2. Enter an absolute expiration date in the form of yy/ddd.

84/005 file will expire on 5 January 1984

82/042 file will expire on 11 February 1982

3. Enter the number of days that the file is to be retained in the form of 0 to 9999.

9 file will expire 9 days after creation.

27 file will expire 27 days after creation.

9999 is used to specify a permanent file.

file serial number	This optional entry allows the user to specify the volume serial number of the first or only reel of tape in file.
volume sequence number	If desired on input files you may wish to use this optional operand to access some volume of a multivolume other than the first. Enter 1 to 4 numeric characters. (This operand is usually omitted)
file sequence number	This optional entry allows the user to access a specific file within a multifile volume. Enter 1 to 4 numeric characters.

Generation number	Optional operand for the user to specify a generation number. Enter 1 to 4 numeric characters.
Version number	Optional operand to sub-identify a particular generation. Enter 1 or 2 numeric characters.

```
// LBLTYP  TAPE        or      // LBLTYP  NSD(02)
```

The LBLTYP statement (reserve storage for label processing) defines the amount of main storage to be reserved at linkage-edit time.

FORMAT

```
// LBLTYP   TAPE
            NSD(nn)
```

TAPE	Used only if tape files requiring label information are to be processed, and no nonsequential DASD files are to be processed.
NSD(nn)	Used if any nonsequential DASD files are to be processed regardless of other type files to be used. nn specifies the largest number of extents to be used in a single file.

The amount of storage that must be reserved for label information is:

1. For standard tape labels 80 bytes.
2. For sequential disk labels NONE.
3. For ISAM or DA disk file labels 84 bytes plus 20 bytes per extent.

EXERCISES

1. Write the JCL to execute a program that is cataloged in the core image library under the phase-name 'WS342' and contains the following select statements:

```
     SELECT CARD-FILE
          ASSIGN TO SYS008-UR-2540R-S.
     SELECT TAPE-FILE
          ASSIGN TO SYS011-UT-2400-S.
```

TAPE-FILE is an output file. Utilize 'ERROR FILE TAPE' as a unique file ID and put the file on tape SN 772206. Retain the file for one year.

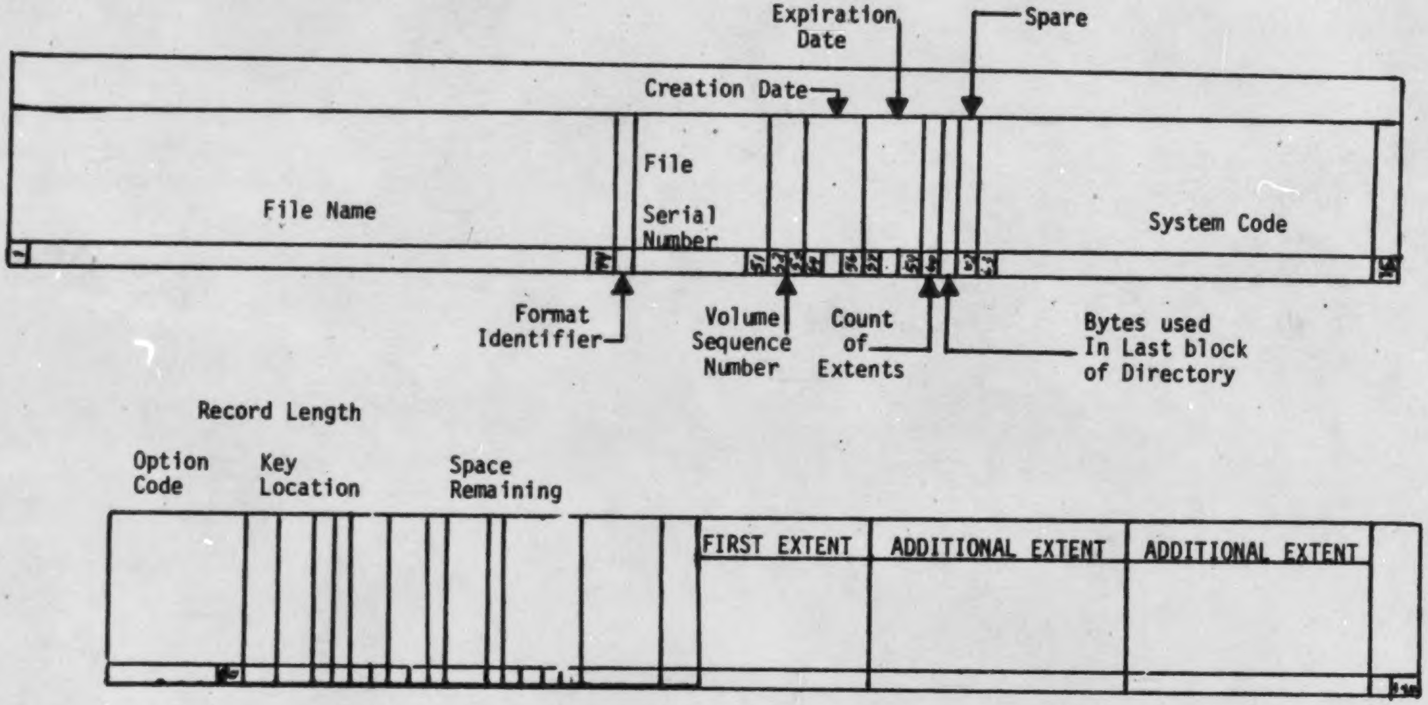
2. Write the ASSGN's and TLBL's for a program containing the following Select statements:

```
     SELECT TAPE-INPUT-FILE
          ASSIGN TO SYS011-UT-2400-S.
     SELECT TAPE-OUTPUT-FILE
          ASSIGN TO SYS012-UT-2400-S-NEWMST.
```

TAPE-INPUT-FILE has no unique file ID but is on tape SN 750349.

TAPE-OUTPUT-FILE is to go on tape SN 781433. Assign your own file ID. The file is to be retained indefinitely.

STANDARD MASS STORAGE DEVICE LABELS -- (VTOC)


Expiration Date
Spare
Creation Date
File Name
File Serial Number
System Code
Format Identifier
Volume Sequence Number
Count of Extents
Bytes used In Last block of Directory
Record Length
Option Code
Key Location
Space Remaining
FIRST EXTENT
ADDITIONAL EXTENT
ADDITIONAL EXTENT

```
// DLBL  DISKFLE,'DISK MASTER FILE',87/130,SD
```

The disk label card is used to provide part of the label information for disk label writing and checking.

FORMAT

```
// DLBL  filename,['file id'],[date], [code]
```

filename This operand serves to link to DLBL card to the COBOL program file description (FD). It consists of 1 to 7 alphameric characters, the first of which must be alpha.

```
EXAMPLE;          SELECT  OM-DISK-MASTER-FILE
                      ASSIGN TO SYS007-UT-2314-S-DISK5.

                  // ASSGN  SYS007,X'132'
                  // DLBL   DISK5,'Mast........
```

file id This operand links the DLBL card to the data set on the disk pack. It is 1 to 44 characters in length the first of which must be alpha. The file id must be unique on any given disk pack.

date The date is used when creating an output file and can have one of two formats;

yy/ddd using year and julian date to specify expiration date.
dddd to specify the number of days the file is to be retained.

code This operand tells the system what kind of file this is;

code	definition
SD	Sequential disk file
DA	Direct access file
ISC	Indexed sequential (used only when creating file)
ISE	Indexec sequential (used for all other activities)

```
// EXTENT  SYS007,123321,1,0,123,1520
```

The disk extent(area) card is used to tell the system such things as what disk pack the file is on, where on that pack does the file begin and how many tracks does the file occupy.

FORMAT

```
// EXTENT  [symbolic unit],[serial number],[type],[sequence number],
           [relative track],[number of tracks]
```

530

symbolic unit This operand must match the symbolic unit in its associated assign card.

serial number Volume serial number of the disk pack for which this extent is effective.

type This operand specifies the type of extent this is;

TYPE	PURPOSE
1	Prime data area for any of the 3 file types.
2	Independent overflow for ISAM files.
4	Index area for ISAM.

sequence number For sequential disk files the first extent is sequence number 0 , the second is 1, the third is 2 and so on.

For ISAM files the following rules apply;

0	Master index only
1	Cylinder index only
2	first prime data extent
3	second prime data extent
.	etc...... 4,5,6....N
N+1	Independent overflow

relative track This operand indicates the starting track of the file relative to the first track on the disk pack.

The user will normally know the cylinder and track number. If so, you will have to convert it to relative track using the following formula;

relative track (RT) = 20 X (cylinder number) + (track number)

EXAMPLE: Your sequential disk file begins on cylinder number 15 track number 5. What is the relative track?

RT = 20 X 15 + 5
RT = 305

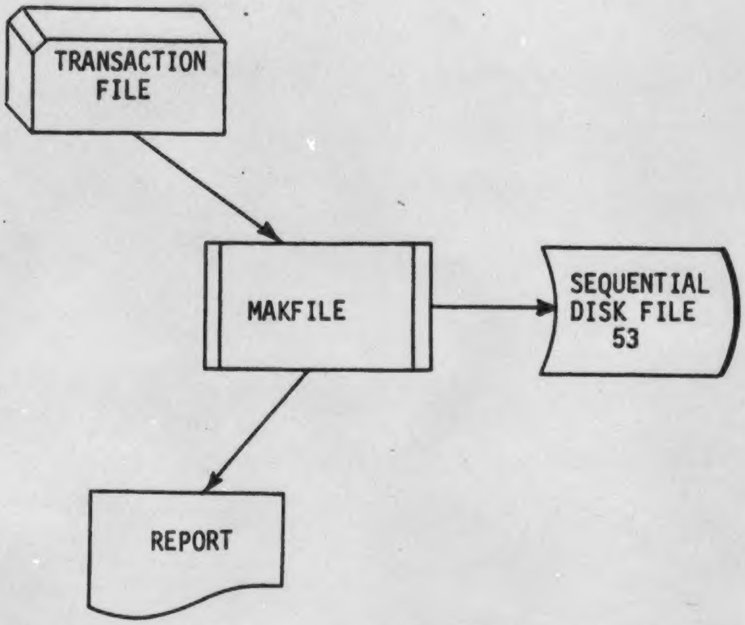
NOTE: For ISAM files prime data extents, the relative track must be a multiple of 20.

number of tracks This operand is the total number of tracks in this extent.

EXAMPLE: Your sequential disk file begins on cylinder 15 track 5 and ends on cylinder 21 track 15. What is the number of tracks?

RT(start) = 20 X 15 + 5 = 305, RT(end) = 20 X 21 + 15 = 435
number of tracks = RT(end) - RT(start) + 1 435-305+1=131

Requirement; You are to write the job control language to execute the program called MAKFILE, which is currently cataloged in the core image library.



The sequential disk file is to be called SEQUENTIAL FILE 53 and is to be placed on disk pack number 555663 which is to be mounted on disk drive 132. The file begins on cylinder 105 track 3 and ends on cylinder 112 track 7. The file is to be retained for 20 days. The COBOL program selects are as follows;

```
FILE-CONTROL.
    SELECT  IT-CARD-FILE
        ASSIGN TO SYS004-UR-2540R-S.
    SELECT  OR-MAKFILE-REPORT
        ASSIGN TO SYS005-UR-1403-S.
    SELECT  OM-DISK-FILE
        ASSIGN TO SYS010-DA-2314-S-SEQFLE.
```

532

SOLUTION:

```
// JOB  EXSEQ

// OPTION  LOG,DUMP

// ASSGN  SYS004,X'00C'

// ASSGN  SYS005,X'00E'

// ASSGN  SYS010,X'132'

// DLBL  SEQFLE,'SEQUENTIAL FILE 53',20,SD

// EXTENT  SYS010,555663,1,0,2103,145

// EXEC  MAKFILE

       INPUT DATA
         DECK

/*

/&
```

RT(start) = 20 X 105 + 3 = 2103

RT(end) = 20 X 112 + 7 = 2247

Number of tracks = 2247 - 2103 + 1 = 145

EXERCISES

1. Write the JCL for a program called 'MAKSEQ' that reads cards and creates a sequential disk file. The file is to be stored on disk pack 122013 which will be mounted on drive 134. It will be stored from cylinder 10 track 0 through cylinder 119 track 19. The file should expire 150 days after it is created. File-ID is CUSTOMER MASTER FILE.

```
SELECT CUST-MASTER-FILE
       ASSIGN TO SYS012-UT-2314-S.
SELECT CARD-INPUT-FILE
       ASSIGN TO SYS005-UR-2540R-S.
```

2. Write the JCL for a card to disk program. The COBOL program is to be compiled and executed. The file is to be stored on disk pack 506012 which will be mounted on drive 132. It will be stored from cylinder 24 track 0 through cylinder 49 track 19. The file-ID is PRICE-MASTER-FILE. The file is to be retained indefinitely.

```
SELECT CARD-MASTER-FILE
       ASSIGN TO SYS026-UR-2540R-S.
SELECT PRICE-MASTER-FILE
       ASSIGN TO SYS024-UT-2314-S-PRIMAST.
```

COBOL PE2/PE3 JCL EXERCISE

COMPILE J01
EXECUTE J01

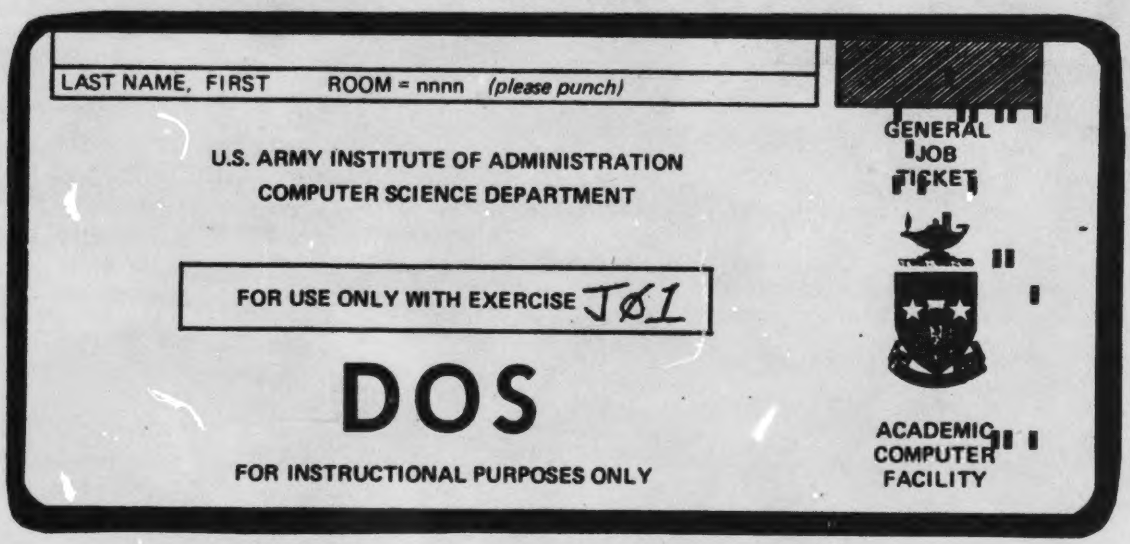
I. REQUIREMENT. Write the DOS JCL necessary to execute your COBOL PE number 2 or 3. You are expected to get the same compiler outputs as before as well as a clean execution of your program.

II. COBOL program changes. There are some differences between DOS and OS COBOL. In order to get a clean compile you will have to do the following:

1. Change the disk file assign to read
   ASSIGN TO SYS006-UT-2314-S.
2. Change the printer file assign to read
   ASSIGN TO SYS005-UR-1403-S.
3. Change the printer FD label clause to read
   LABEL RECORDS ARE OMITTED
4. Remove the printer FD BLOCK CONTAINS clause.

III. METHODS.

1. Replace the // JOB name card with the following card filled out the same way as your normal computer run card;



LAST NAME, FIRST ROOM = nnnn *(please punch)*

GENERAL JOB TICKET

U.S. ARMY INSTITUTE OF ADMINISTRATION
COMPUTER SCIENCE DEPARTMENT

FOR USE ONLY WITH EXERCISE J01

DOS

FOR INSTRUCTIONAL PURPOSES ONLY

ACADEMIC COMPUTER FACILITY

2. Be sure all of your JCL is listed on the printer.
3. Do not use the DECK or CATAL optins.
4. The disk pack is to be mounted on disk drive 'DK3'.
5. The output is to be printed on printer 'PR0'.
6. The serial number of the disk pack is DOSWK3.
7. The file ID is 'SOFTWARE DATA FILE 14 * RESIDENT'.

INDEXED SEQUENTIAL ACCESS METHOD (ISAM) FILES

An ISAM file is basically a sequential file with a series of indexes that allow the processing of single records located anywhere within the file. An ISAM file can also be processed sequentially if desired. There are several different types of areas associated with an ISAM file and these are explained below (See diagram on the next page as you read).

PRIME AREA is where the records are located and may occupy one cylinder or many disk packs. The prime area is defined by the programmer thru JCL..The records are located in this area in sequential order (logically) by record key in ascending order. A KEY is some unique field within the logical record. In a personnel file perhaps we would use Social Security Number.

INDEX AREA is the place on the disk pack where ISAM builds and maintains a series of indexes used to access or add records to the file when processing in a random mode.

TRACK INDEX occupies the first track of each cylinder of the prime area. The track index is defined by ISAM and as with all other indexes is loaded by ISAM. Its' function is to tell ISAM on which track of this cylinder the record is located or if being added, on which track it is to be added to.

CYLINDER INDEX is defined by the programmer thru JCL. Its' function is to tell ISAM on which cylinder the record should be located.

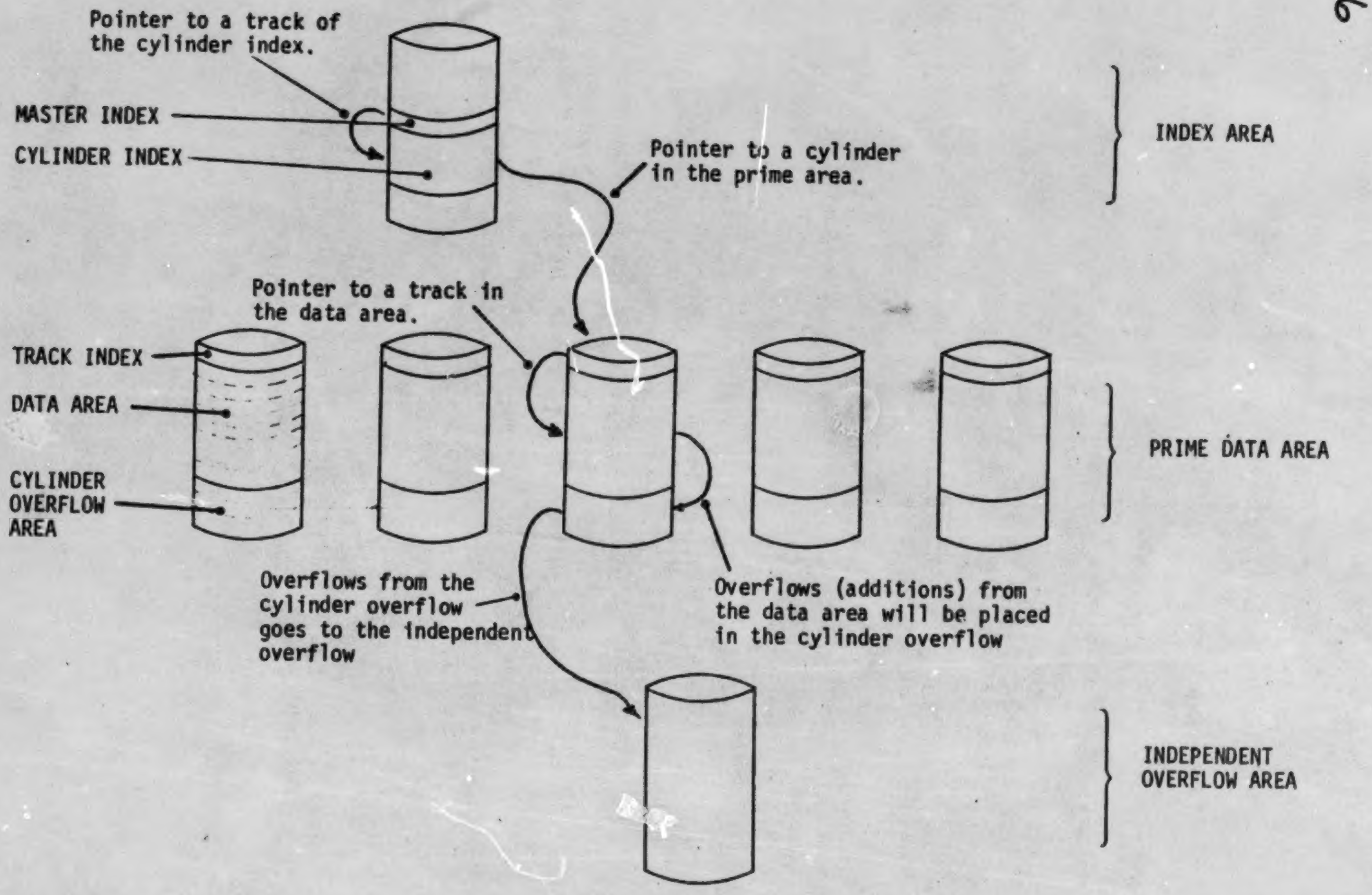
MASTER INDEX is defined by the programmer using JCL. Its' function is to tell ISAM which track of the cylinder index to look for this particular record. The master index is the only optional index in the ISAM file. It is normally defined if the cylinder index is greater than 4 tracks.

OVERFLOW AREA is used when records are added to an already established file.

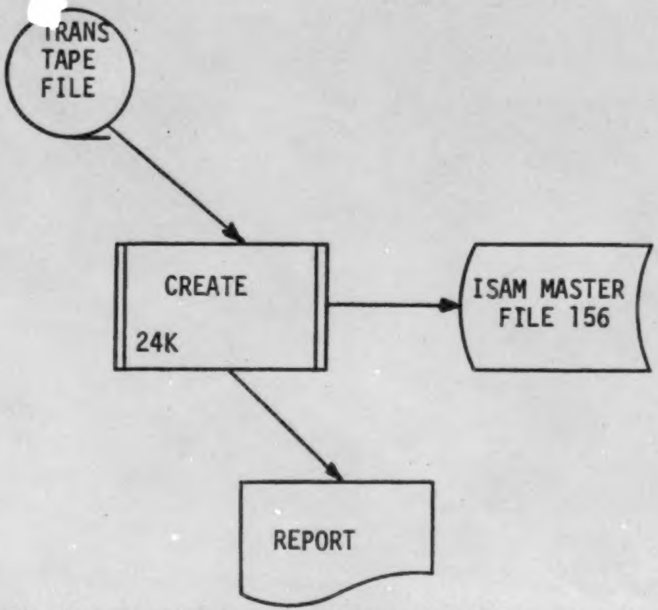
CYLINDER OVERFLOW. A certain number of whole tracks are reserved in each cylinder for overflow records from the prime area. The programmer may specify the number of tracks to be reserved by means of the APPLY CYL-OVERFLOW clause. If zero is specified no cylinder overflow will be reserved. If the clause is omitted, ISAM defaults to 20% of each cylinder.This area is not defined by JCL.

INDEPENDENT OVERFLOW AREA. Overflow records from anywhere in the prime area are placed in a certain number of cylinders reserved soley for this purpose. The size and location of this area is defined by the programmer thru JCL.

ELEMENTS OF AN INDEXED SEQUENTIAL (ISAM) FILE


Pointer to a track of the cylinder index.
MASTER INDEX
CYLINDER INDEX
Pointer to a cylinder in the prime area.
INDEX AREA
Pointer to a track in the data area.
TRACK INDEX
DATA AREA
CYLINDER OVERFLOW AREA
PRIME DATA AREA
Overflows from the cylinder overflow goes to the independent overflow
Overflows (additions) from the data area will be placed in the cylinder overflow
INDEPENDENT OVERFLOW AREA

Requirement: You are to write the job control language needed to execute the program called CREATE, which is currently cataloged in the core image library. The systems flow chart below will assist you by showing you the relationships of all files.



The transaction tape file has a file Id of 'TRANS TAPE 156' and the file serial number is 556783. It is to be mounted on drive 182.

The ISAM file occupies 3 disk packs as follows;

The MASTER INDEX is on disk pack number 122375 and begins on cylinder 15 track 3 and ends on cylinder 15 track 5. Mount pack on drive 131.

The CYLINDER INDEX,as required,begins on the next track after the master index and takes 12 tracks.

The PRIME AREA is on disk pack number 155732 and occupies cylinders 5 thru 167. Mount that pack on drive 132.

The INDEPENDENT OVERFLOW AREA is on disk pack number 123723 and occupies all of cylinder 12. Mount that pack on drive 133.

Retain the file for 180 days and call it 'ISAM MASTER FILE 156'.

The COBOL program selects are as follows;

```
FILE-CONTROL.
    SELECT IT-TRANS-TAPE-FILE
        ASSIGN TO SYS010-UT-2400-S-SQTAPE.
    SELECT OM-MASTER-FILE
        ASSIGN TO SYS015-DA-2314-I-ISMAST
        ACCESS IS SEQUENTIAL
        RECORD KEY IS OM01-SSAN.
    SELECT OR-MASTER-CREATE-RPT
        ASSIGN TO SYS005-UR-1403-S.
I-O-CONTROL.
    APPLY CYL-OVERFLOW OF 2 TRACKS ON 0M-MASTER-FILE
```

SOLUTION:

```
// JOB  ISAMEX
// OPTION  LOG,DUMP
// ASSGN  SYS005,X'00E'
// ASSGN  SYS010,X'182'
// TLBL  SQTAPE,'TRANS TAPE 156',,556783
// ASSGN  SYS015,X'131'
// ASSGN  SYS020,X'132'
// ASSGN  SYS025,X'133'
// DLBL  ISMAST,'ISAM MASTER FILE 156',180,ISC
// EXTENT  SYS015,122375,4,0,303,3
// EXTENT  SYS015,122375,4,1,306,12
// EXTENT  SYS020,155732,1,2,100,3260
// EXTENT  SYS025,123723,2,3,240,20
// EXEC  CREATE
/&
```

EXERCISES

1. Write the DLBL and EXTENT cards to create an ISAM file. The prime data area will be the area from cylinder 20 track 0 through cylinder 119 track 19. The master index will use cylinder 140 track 0 through cylinder 140 track 3. The cylinder index will use cylinder 140 track 4 through cylinder 140 track 15. The file is to be stored on pack 223556 and is to be saved for 500 days. Use the following select statement.

```
SELECT INVENTORY-MASTER-FILE
           ASSIGN TO SYS009-DA-2314-I-MSTFLE.
```

2. Write the ASSGN, DLBL, AND EXTENT cards to create an ISAM file on disk pack 443556 drive 135. The file should not expire. The prime data area will be cylinder 29 track 0 through cylinder 35 track 19 and the cylinder index will be from cylinder 28 track 0 through cylinder 28 track 2. Use the following select statement.

```
SELECT CUSTOMER-MASTER
           ASSIGN TO SYS005-DA-2314-I-CUSMSTR.
```

EXERCISES

3. Write the JCL to compile and execute a COBOL program which is to randomly retrieve and update the master file created in problem #2. The file-ID for the transaction file is TRANSACTION FILE and the serial number is 100005. No other information is needed for checking this label. Disk drive 134 and Tape drive 183 are to be used. Use the following COBOL select statements.

```
SELECT MASTER-FILE
       ASSIGN TO SYS021-DA-2314-I-CUSMSTR.
SELECT TRANS-FILE
       ASSIGN TO SYS022-UT-2400-S-TRNSFLE.
```

LIBRARIAN

As we have seen the DOS system consists of three libraries; the core image library, the relocatable library, and the source statement library. We have also seen some of the functions performed on the libraries.

1. The linkage editor places the link edited phases into the core image library either in the temporary portion for immediate execution or catalog them as permanent members for execution at any time.
2. The system loader finds executable phases in the core image library and loads them into memory for execution.
3. Link editor will load modules into SYSLNK from the relocatable library to be link edited.
4. The linkage editor will extract relocatable IOCS modules from the relocatable library and attach them (link edit) to COBOL Programs.
5. The COBOL compiler and the assembler have the capability of extracting source code from the Source statement library and combining it with other source code.

FUNCTIONS OF LIBRARIAN

1. Maintenance functions add, delete, or rename components of the three libraries, condense or remove spaces that may exist in the directories or libraries themselves. Alter the amount of space allocated for each library. The program phase that performs this maintenance function is called MAINT and it applies to all three libraries.

2. Service functions will print and/or punch library members and directories. There all four program phases that perform these services and they are:

DSERV - library directories (all)
CSERV - core image library
RSERV - relocatable library
SSERV - Source statement library

3. Copy functions will either completely or selectively; copy the disk pack on which the system resides, create private libraries, or merge libraries. The program phase that copies libraries is called CORGZ.

DISPLAY (SERVICE FUNCTION)

1. DSERV: the purpose of DSERV is to list, on the printer, the contents of the directories of the three DOS libraries in a readable format.

Format of control card:

```
b̸ DSPLY directory [ ,directory]
```

directory: can be one of the following:

TD

CD

RD

SD

ALL

EXAMPLE: The following example would be used to cause the system to print out the directory for the core image library.

```
// JOB CORING
// EXEC DSERV
b̸ DSPLY CD
/*
/&
```

544

```
CORE IMAGE DIRECTORY          24/01/78                                   PAGE   10

                   PHASE      DISK       NO.   SIZE        LOAD     ENTRY
                   NAME       ADDR       RCDS LAST RCD     ADDR     ADDR

                            -----------DEC---------     -----HEX------

                               C  H  R
                   FLCRSEQE  034 18 03   001   0288      005000   005000
                   FLCRKONE  034 18 04   001   0704      005000   005000
                   U43ATP    034 19 01   007   0720      000000   000608
                   P23ATPLE  035 00 04   001   1601      000000   0005C0
                   P23SSPLE  035 01 01   007   0209      000648   002148
                   P23SPWLE  035 02 04   020   0205      002E80   00A6D8
                   PW4140K1  035 07 04   018   0409      000000   0068D8
                   P23ATP    035 12 02   001   1378      0360C0   0360C0
                   SUBSPOOL  035 12 03   006   1659      036628   038128
                   SUBPOWER  035 14 01   015   0884      038DA0   03D7F8
                   P23ATP30  035 17 04   001   1364      0360C0   0360C0
                   P23SSP30  035 18 01   006   1659      036618   038118
                   P23SPW30  035 19 03   015   0852      038D90   038D90
                   P23ATP40  036 03 02   001   1641      000000   0005E8
                   P23SSP40  036 03 03   007   0209      000670   002170
                   P23SPW40  036 05 02   018   0401      002ED8   009AA8
                   P20ATS    036 09 04   020   0912      000000   006538
                   $262DAP5  036 14 04   013   0218      000000   000000
                   *262DEU4  036 18 01   013   0218      000000   000000
                   $262DJA8  037 01 02   013   0218      000000   000000
                   *262EBH6  037 04 03   013   0218      000000   000000
                   *262LEE2  037 07 04   019   0298      000000   000000
                   $262SJA8  037 15 04   013   0218      000000   000000
                   $262TEP3  037 19 01   019   0298      000000   000000
root phase         $262TJA3  038 03 04   013   0218      000000   000000
of COBOL           *262TRI4  038 07 01   013   0218      000000   000000
compiler           P04ATP    038 10 02   006   1659      000000   001800
                   FCOBOL    038 11 04   017   0650      007800   008470
                   FCOBOL10  038 16 01   017   1428      00A360   00A36A
                   FCOBOL12  039 00 02   019   1216      00A360   00A36A
                   FCOBOL11  039 05 01   016   0140      00A360   00ABD8
                   FCOBOL20  039 09 01   013   1264      00A158   00A168
                   FCOBOL22  039 12 02   015   0384      00A158   00B148
see page 8-7       FCOBOL21  039 16 01   016   0404      00A158   00A9D0
                   FCOBOL30  040 00 01   007   1540      00A158   00C7A8
                   FCOBOL40  040 01 04   023   1583      009DA0   012474
                   FCOBOL50  040 07 03   022   0048      009DA0   011A70
                   FCOBOL51  040 13 01   021   0852      009DA0   0113B0
                   FCOBOL60  040 18 02   016   1426      009DA0   009DAC
                   FCOBOL61  041 02 02   002   0573      009DA0   009D84
                   FCOBOL70  041 02 04   026   0392      009288   009294
                   $$AA2J03  041 09 02   001   0002      000000   000000
supervisor         *262SAP4  041 09 03   013   0219      000000   000000
transient          *JOBCTLA  041 12 04   004   0408      000000   000FA8
routine            *JOBCTLD  041 13 04   004   0944      000FA8   000FA8
                   *JOBCTLF  041 14 04   004   0453      000FA8   000FA8
                   *JOBCTLG  041 15 04   004   0744      000FA8   000FA8
                   *JOBCTLJ  041 16 04   004   0406      000FA8   000FA8
```

```
SYSTEM DIRECTORY                    CORE-IMAGE                  RELOCATABLE                 SOURCE-STATEMENT

24/01/78                            -----------------------------DECIMAL----------------------------------

                                     C  H  R  E                   C  H  R  E                  C  H  R  E
DIRECTORY STARTING ADDRESS           00 10 01                     46 00 01                    61 00 01
DIRECTORY NEXT ENTRY                 00 13 11 04                  46 01 14 15                 61 00 20 09
DIRECTORY LAST ENTRY                 00 15 15 17                  46 04 17 19                 61 01 27 09

LIBRARY   STARTING ADDRESS           00 16 01                     46 05 01                    61 02 01
LIBRARY   NEXT AVAILABLE ENTRY       42 11 01                     57 06 02                    88 14 15
LIBRARY   LAST AVAILABLE ENTRY       45 19 04                     60 19 16                    98 19 27

                                    -----------------------STATUS  INFORMATION----------------------------

DIRECTORY ENTRIES ACTIVE             975                          609                         187

LIBRARY   BLOCKS ALLOCATED          3616                         4720                       20466
LIBRARY   BLOCKS ACTIVE             3288                         3357                       14373
LIBRARY   BLOCKS DELETED              52                          180                         545
LIBRARY   BLOCKS AVAILABLE           276                         1183                        5548

AUTOMATIC CONDENSE LIMIT              00                           00                          00

LIBRARY   ALLOCATED CYLINDERS         45                           15                          38
DIRECTORY ALLOCATED TRACKS            06                           05                          02
```

CSERV: The purpose of CSERV is to print out on the printer and/or punch on the card punch the members of the core image library.

Format of control card:

```
ƀ operation  ƀ operand [ , operand 2]
```

Operation: May be one of the following:

DSPLY
PUNCH
DSPCH

Operand: May be one of the following:

phasename
progl.ALL
ALL

EXAMPLE: Print out all of the phases of FCOBOL

```
// JOB PRTFCOB
// EXEC  CSERV
ƀ DSPLY FCOB.ALL
/*
/&
```

```
                                                 CORE IMAGE LIBRARY

                                  PHASE FCOBOL30   LENGTH -011,668 BYTES.

00A150                              C6C3D6C2 D6D3F3F0   F3F687EA D5011002 10064780 87849500   10054770 87EA9513
00A180   10061002 474087FA 920098B2 D20098B3   100317AA 43A10002 89A00006 42A098B4   D60098B4 1005D200
00A1B0   89A00006 42A098B6 D60098B6 1009D201   91A598B3 D20191AD 98359201 91A7922B   91AF0A09 92F198B2
00A1E0   47F087D8 92F398B2 47F087D8 92F498B2   47F087D8 92F598B2 47F087D8 D2008C46   98B2D203 8C538DCE
00A210   92F08631 96F08631 47F0874A 92F29844   92F08631 47F0874A 92F39844 92F08631   47F0874A 92F49844
00A240   92F59844 92F08631 47F0874A 92F69844   92F08631 47F0874A 4120E0CC 58709798   D2077068 2010D207
00A270   2008D207 71942018 4120E0EC 587097A0   D2077068 2010D207 709C2000 D20770D0   2008D207 71942018
00A2A0   45508 90C D2004041 9151D201 40439152   92064020 41108D1E 0A009180 10024710   88CA0A07 91401005
00A2D0   F0009180 8D8347E0 88F6D203 91A891A4   58108DD2 92FF1000 D23C1001 10004160   E018D73D 10006030
00A300   E01447F0 841041C0 706841B0 E0404120   E0814530 894C41C0 709C41B0 E0504120   E08F4530 894C41C0
00A330   E0FD4530 894C41C0 719441B0 E06C1822   4530894C 07F5D503 C000916C 0783D200   4041C001 00000000
00A360   C0DEC0DE 91000000 0000A2A8 0000A52C   0000A4CC 0000A424 0000A421 0000A380   0C0000AC 00F0007A
00A390   00000000 00000000 00000000 00000000   00000000 00000000 00000000 18211811   D2008D12 00000000
00A3C0   91581B21 18114310 40435C00 91581A21   43004044 06001A20 41100200 19214720   883C0620 5A20907C
00A3F0   95018000 4770BA12 92C0B000 92054020   0A009180 00000209 000001FF 003FFF00   00000200 00000200
00A420   000058E1 400000FF 0000A658 003FFFFF   FF000000 00FFFFFF D5002000 90000200   00000000 00000000
00A450   C00000FF D2002000 8000D200 20009000   00000000 50C0F064 58C0F060 90E1C290   5810C150 D207C154
00A480   D203C1F0 11C89101 11404710 F0549201   11405810 114041F0 0006D7FE 10001000   411100FF 46E0F036
00A4B0   C2185BF0 C15405FF 90E1C290 58C0F064   07FF0000 0000A164 00000000 9037C22C   90FFC240 18525450
00A4E0   00095C40 C2285870 C1504177 00AC5877   00005877 00004135 70009180 30004780   F04C9160 30004710
00A510   C22405FF 98EFC240 58030000 5400C2D0   5420C2A4 1A209837 C22C07FE 5090C248   41A00004 18881B00
00A540   18901B88 5D80C2A0 41A88000 1AA85880   C1505888 014041AA 8000D202 C28DA000   5880C2BC 5890C248
00A570   58C0F218 90FFC290 91081000 4780F078   904BC258 18669104 10004770 F0405820   C2DC5850 C2E01255
00A5A0   4780F048 41E0000C 47F0F1B6 58210004   5020C2DC 58F0C20C 05EF58F0 C2949180   20004780 F05F4122
00A5D0   F038D202 C2852000 5850C2B4 5450C2C8   47F0F088 91041000 4710F1CC 904BC258   5860C2C4 18335891
00A600   58F0C208 05EF58F0 C29412B8 4770F0AC   41E00004 47F0F1B6 12664770 F08A5980   C2DC4740 F1281828
00A630   C2941B77 91802000 4780F0E2 D202C2B5   20005870 C2845470 C2C84122 00031266   07761985 4780F11C
00A660   F0FC18A2 41220003 4480C2D4 4770F11C   12334780 F11241F0 000847F0 F1B61838   184AD202 C2B8A000
00A690   47F0F0AC 12334770 F13641E0 000447F0   F1B61823 58F0C20C 05EF58F0 C29491C0   20004780 F1564122
00A6C0   00034122 B0019110 10004780 F1701244   4770F186 41F0000C 47F0F1B6 910C1000   4780F19C 12444770
00A6F0   F19ED202 C2B5C2B8 5810C2B4 5410C2C8   5010C2E0 47F0F1A2 18445040 C2E05030   C2DC984B C2585BE0
00A720   07FE50E0 F21C984B C2585BE0 C29058C0   F22058F0 F21C07FE 5080C258 58210000   5420C2D0 5020C2DC
00A750   C29491C0 20004780 F1FA4122 00034740   F1FA4122 00031888 5080C2E0 43820000   41228001 5880C258
00A780   07FE0000 0000A164 00000000 0000881C   50C0F0EC 58C0F0E8 90EFC290 5040C258   182158F0 C20C05EF
00A7B0   5A00C2AC 91C02000 4780F038 41220003   4740F038 41220003 18444342 00004124   20014342 000091F0
00A7E0   20004710 F05A4122 00014122 00064342   00004122 400147F0 F08C959A 20004780   F07A959A 20004770
00A810   5A40C1F8 88400004 41224000 19204780   F09A91FF 20004770 F0825410 C2A84111   02001821 58F0C20C
00A840   F08A1832 5830C2B4 1A1391C0 20004780   F0CE4122 00034740 F0CF4122 00031832   43420000 41224001
00A870   58C0F0EC 07FE0000 0000A164 40000143   58108C48 05010200 500090DC 47F07014   0000C828 45E08A88
00A8A0   954490E2 478070A8 958190E2 47807060   950590E2 47807050 952290E2 47807050   952490E2 47807050
```

8-7

CSD.SFW.OS

547

RSERV: Performs the print and punch functions for the relocatable library.

Format of control card

```
ƀ operation ƀ operand1 [ ,operand2]
```

Operation:

```
DSPLY

PUNCH

DSPCH
```

Operand:

```
Modulename

proql.ALL

ALL
```

EXAMPLE: Punch out the printer IOCS module called IJDFAPIZ

```
// JOB PRTPTR

// EXEC RSERV

ƀ PUNCH IJDFAPIZ

/*

/&
```

```
IJDFAPIZ  V.M      3.9                        00003 BLOCKS        SYSTEM  RELOCATABLE LIBRARY
          ESD               032   0001   IJDFAPIZ  SD  000000  000118       IJDFAZIZ  LD  000000  000001
          TXT      000000   136   0001   0A320000 47F0F084 0A320000 47F0F01A C9D1C4C6 C1D7C9F9 F3F99180
                                         F0240AC7 90CEF0EC 58E01018 06E0D200 1017E000 18CC18DC 43C01017
                                         43DEF0F7 19DC4780 F05046F0 F0400A32 43DEF107 43C01016 19CE4770
                                         10164780 FC769208 10280A00 91801002 4710F076 0A074200 10284200
                                         91801002 4710F08A
          TXT      000088   136   0001   0AC7D201 10261003 92011028 58E01028 D2021029 101950E0 101806F0
                                         44001022 58E0F0F4 0A0007FE 91801002 4710F08E 0A079101 10154780
                                         10159102 1026078E 47F0F0DA 91011027 078E1200 4780F0F4 18F007FE
                                         0A0007FE C0000000 00000000 00000000 F2C2C1F9 F8F7F6F5 F4F3C3F1
                                         93C8D3C8 03BBB3AB
          TXT      000110   008   0001   A398E388 03181308
          END      404040
```

8-9

CSD.SFW.OS

550

SSERV: Performs the print and punch functions for the source statement library. The source statement library is divided into parts called sublibraries. There is one for COBOL called 'C' and one for the assembler called 'A' and in the case of OSAIA one called 'P' for cataloged JCL Books.

Format of control card:

b̸ operation b̸ operand1 [, operand2]

Operation;

DSPLY

PUNCH

DSPCH

Operand; sublib.book

sublib.ALL

ALL

Example: Print and punch the assembler sublibrary of the source statement library.

```
//JOB SORLIB
// EXEC SSERV
b̸  DSPCH  A.ALL
/*
/&
```

```
V.M 0.0            18 BLOCKS            PRIVATE SOURCE-STATEMENT LIBRARY

         BKEND   P.C24B

* $$/*
// EXEC LNKEDT
// ASSGN SYS006,X'CRO'
//ASSGN SYS005,X'PRO'
// EXEC
CSM147206
SGM145204
1SG137193
MSG135191
SP7118181
SFC116179
SB6108168
SSG106166
SP5105156
SGT103154
SP4082136
PFC080116
PV207116
PV1067116
* $$/*
255656008FOX HENRY E           0101SGM101639264
326366478ISSAACS JUDY I        0101SSG060924487
353120375BERGERT KENNETH B     0101PV1050161948
545508335CROMBAUGH SUSAN C     0101SGM061676629
631608320ESSELBORN RAYMOND     0101CPL080514535
631985218KAZACOFF KOTCHO K     0101SGM051557952
639116773GRIEBELBAUER JOHN     0101SP6001041329
649788627JOHNSON SAM J JR      0101SGT050773699
787425848LEWIS FLOYD L         0101SGC111139387
846775121ADOMATIS RUSSELL      0101E-4130614992
848521418DEZUTTI LOUIS D JR    0101SSG030963552
888124925HOLTZMAN SALLY H      0101PFC040369948
251745252QUINCY WALTER Q       0102SO6011054487
255973250OXLEY MORRIS O JR     0102PFC000206662
356879414MCCARTHY MARY M       0102SP6011054487
547083118SMITH PITTMAN S       0102PFC000206662
637427078UNDERWOOD DEVIL H     0102SP6081005936
637474514VERMILLION BETTY V    0102)V2010213992
773137376RIVERA FRANCES R      0102SGT090728487
786214360WHEELER GREGORY W     0102PV1000058436
831195644TOMLINSON HARRY T     0102PV1020303236
849238650NEWTON DWIGHT N       0102SSG031019192
846493958PARKER CLYDE B        0102E-5100832938
248041032XAVER RUTH X          0102PV2040435336
291777069ZUCKERBURG HARRY Z    0102SSG131051363
353865446YATES CLARENCE Y      0103PV1050025653
115678395GRIFFIN ARCHIE G      0103PV1100010593
161312570COSTELLO LOUIS C      0103SP6131040636
337255981DIRKSON EVERETTE D    0201CPL000405288
535973792ABBOTT BUD A JR       0201SSG131051363
569264882FORD GERALD F         0103PV1050025653
737745671HUMPHREY HUBERT H     0103PV1100010593
```

CSD.SFW.OS

```
778712763EAGLETON THOMAS E        021SGT080624264
825391253ISLEY BROTHERS I         0201E-2010364888
923948125BUTZ EARLE B             0201E-7071187635
263233736JOHNSON LYNDON J         0301PV1090206487
357189154MANSFIELD MIKE M         0301SFC011117629
66588495KISSINGER HENRY K         03011SG021665787
762042553NIXON RICHARD N          0301SSG050852753
297864442LINCOLN ABRAHAM L        0301MSG101573938
384353898QUEENIE PATRICIA         0302SP5020646935
437293380VERDRIVE BACHMAN         0302SP6070902498
654619466PATTON GEORGE P          0302SGT020898236
797833538BUMSFELD DONALD R        03020PL160482498
062535910STEPPENWOLFE CRAZE       0303SP4010465745
297275429TRUMAN HARRY T           030PFC000361248
196468460UNDERDOG CANINE U        0304PV2090315236
258492158ZIMMERMAN JAMES Z        0304SSg130989953
328339176WALLACE GEORGE W         0304PV2120025523
384422357XAVIER I A               0304SGT070745888
611989829YANKEE CONNETICUT        0304SP4080448142
846244754VICTORIOUS NOT SO        0304PV1050331888
```

BKEND

CATALOG NEW MEMBERS TO LIBRARIES

CORE IMAGE LIBRARY: We have already seen in our discussion of Job Control that the LINKAGE EDITOR is the program that catalogs phases to this library.

EXAMPLE: Catalog an object deck to the core image library and call the new phase CATX.

```
    // JOB CATLEX

    // OPTION CATAL
    ƀ PHASE  CATX,*
    ƀ INCLUDE

    OBJECT DECK

     /*

     // EXEC LNKEDT

     /&
```

RELOCATABLE LIBRARY: The program that catalogs modules into this library is called MAINT.

Format of control card:

```
ƀ CATALR modulename [ , v.m]
```

Modulename	A unique name consisting of 1 to 8 alphameric characters the first of which is alphabetic.

v.m This optional entry is used to provide a version and modification number to further identify this module from other previous copies with the same name.

v can range from 0 to 127.

m can range from 0 to 255.

EXAMPLE: Catalog an available object module to the relocatable library and call it MOD516. For this copy use this month (June) for the version and todays date (27th) for the modification number.

```
// JOB CATALREX
// EXEC MAINT
b̸ CATALR  MOD516, 6.27
OBJECT MODULE
/*
/&
```

SOURCE STATEMENT LIBRARY: The program that catalogs to this library is called MAINT.

Format of control cards:

b̸CATALS	Sublib.bookname [,v.m. [, C]]
Sublib	The single alphabetic character which identifies which sublibrary this book is to be cataloged into.
bookname	The unique name for this book within its sublibrary. 1-8 alphameric character the first of which must be alphabetic.
v.m.	Further identification to allow us to differentiate this from previous versions.
C	This character is entered as part of the directory and, if used, requires the user to specify the correct version and modification number if he is trying to change the module using the UPDATE feature of MAINT.

b̸ BKEND [sub.book] [,SEQNCE] [,count] [,CMPRSD]

This control card must appear in front and behind of each book that is cataloged into the source statement library. The optional operand entries, if used, need only be put in the front BKEND card.

Sub.book	same as operand in CATALS card
SEQNCE	The sequence numbers punched in card-columns 76 to 80 are checked for ascending sequence numbers.
Count	used to specify the number of cards in the book including the two BKENDs.
CMPRSD	compress card formats by removing blanks.

EXAMPLE: Catalog into the source statement library a standard file description book and call it STDFD74.

```
// JOB CATALGS
// EXEC MAINT
b̸  CATALS  C.STDFD74
b̸  BKEND  C.STDFD74 ,,,CMPRSD

     SOURCE BOOK

b̸  BKEND
/*
/&
```

RENAMING MEMBERS OF LIBRARIES

The program MAINT performs this function for all three libraries.

Core Image Library

```
// JOB EXAMCIL

// EXEC MAINT

b̸ RENAMC oldphasename, newphasename

/*

/&
```

Relocatable library

```
// JOB EXAMRLL

// EXEC MAINT

b̸ RENAMR oldmodulename, newmodulename

/*

/&
```

Source Statement library

```
// JOB EXAMSSL

// EXEC MAINT

b̸ RENAMS sublib.oldbookname, sublib,newbookname

/*

/&
```

NOTE: All of these operations could have been done in a single job.

2. The operand field can be repeated to specify more than one rename operation.

DELETING MEMBERS FROM LIBRARIES

The program MAINT performs this function for all three libraries.

CORE image library

```
                // JOB EXDELCIL
                // EXEC MAINT
                ␢ DELETC  phasename
or              ␢ DELETC  prog.ALL
                /*
                /&
```

NOTE: prog.ALL will delete every phase in the core image library whose name begins with the 4 characters specified.

Relocatable library

```
                // JOB EXDELRLL
                // EXEC MAINT
                ␢  DELETR modulename
or              ␢  DELETR Prog.ALL
or              ␢  DELETR ALL
                /*
                /&
```

NOTE: Prog.ALL deletes all modules whose first 3 characters are specified.
ALL deletes the entire library.

SOURCE STATEMENT LIBRARY

```
                // JOB EXDELSSL
                // EXEC MAINT
                 b̸ DELETS Sublib.bookname
or               b̸ DELETS sublib.ALL
or               b̸ DELETS ALL
                /*
                /&
```

NOTE: Sublib.ALL delets the entire sub-library

ALL deletes the entire library.

CONDENSING LIBRARIES

If a delete is perfomed the reference to the member in the library directory is removed. However, the old copy still occupies space in the library itself.

The same is true if a new member is cataloged with the same name as an old member. The old member is deleted from the directory but still occupies space in the library itself.

The program MAINT perform the condense function for all three libraries.

```
        // JOB EXCONDLB
        // EXEC MAINT
        b̸  CONDS CL
  or    b̸  CONDS RL
  or    ø  CONDS SL
  or    ø  CONDS CL, RL, SC
        /*
        /&
```

UPDATE (Change) SOURCE LIBRARY BOOKS

Once books are cataloged to the source statement library there is no need to maintain them in punched card format. However, there may be a requirement to make corrections or minor changes to library members. One way to do this is to punch up a new copy with changes and catalog it over the old copy. Another method would be to use the UPDATE function of the program MAINT to make these minor changes.

NOTE: In order to update a book it must have been sequenced prior to being cataloged. In the book card columns 73 - 76 will contain the 1st four characters of the bookname and card columns 77 - 80 will contain ascending sequence numbers.

The control cards are as follows:

```
// EXEC MAINT
b̸ UPDATE sublib.bookname, [s.book], [v.m], [nn]
) ADD seq-no
) REP first-seq-no [ ,last-seq no]
) DEL first-seq-no[ , last-seq-no]
) END [v.m.[,C]]
```

```
UPDATE    sublib.bookname ,[S.book] ,[v.m.], [mm]
```

Sublib.bookname	The name of book being updated
s.book	The temporary name given to the old book to preclude deleting it, in the event that an error was made in the update.

NOTE: If the update was successful this temporary book would have to be deleted.

v.m.	An optional that would be required if the old book requires change level verification.
mm.	This is the interval to be used when new book is assigned sequence numbers. Default is 01 if no resequence desired enter "NO" in this position.

```
) ADD  seq-no
```

Seq-No applies to the sequence number in card columns 77 - 80 of member book cards.

ACTION: The 'seq-no' in the ADD card serves to indicate behind which card in the oldbook to place the cards that follow.

EXAMPLE:

OLD BOOK

```
05 Filler              Pic X (10)               SDFD0120
                   Value spaces.                SDFD0130
05 Filler              Pic X (9)                SDFD0140
                   Value 'LAST NAME'.           SDFD0150
```

CONTROL CARDS

```
// JOB UPDATEIT
// EXEC MAINT
   UPDATE   C.SDFD150,,,10
)  ADD 0130
     05 Filler              Pic  X(4)
                         Value 'Dept'.
)  END
/*
/&
```

NEW BOOK

```
05  Filler           Pic  X(10)         SDFD0120
                   Value spaces.        SDFD0130
05  Filler           Pic  X(4)          SDFD0140
                   Value 'Dept'.        SDFD0150
05  Filler           Pic X(9)           SDFD0160
```

564

) REP first-seq-no [, last-seq-no]

ACTION: The cards in the old book starting with 'first-seq-no' and ending with 'last-seq-no' are deleted and the source card(s) that follow the) REP card are put in their place. One or more old cards can be replaced by one or more new cards.

) DEL first-seq-no [, last seq-no]

ACTION: If only the first operand is used, only that card is deleted. If the two operands are used the series of cards beginning with the first and ending with the second will be deleted.

) End [v.m. [,C]]

This card indicates the end of updates to a book.

v.m	new version and modification number to be placed on new book. Suggestion: month day (6.29)
C	indicates change level verification is required before any subsequent updates to this book.

REALLOCATION FUNCTION

The reallocate function is used when it becomes necessary to change the size of the libraries on the system resident pack.

```
// JOB CHNGIT
// EXEC MAINT
b̸ ALLOC id=cylin (track) [, id=cylin (track),....
/*
/&
```

id	this tells the system which library CI (Core image), RL(relocatable), or SL (Source statement).
cylin	Specifies the number of cylinders to be allocated for a particular library.
track	Specifies the number of track to be taken from the beginning of the first cylinder to store the library index.

NOTE: for the computation of required space see IBM system control and service manual.

COPY FUNCTION

The copy function perform the following operations, individually or in combination:

1. Defines and/or creates a new system Pack.
2. Defines and/or creates private libraries.
3. Transfers phases, modules, or books between any assigned libraries.

Define and create new system pack

```
// JOB COPY

// ASSGN SYS002, X' 191"

// DLBL IJSYSRS, 'DOS SYSTEM RESIDENCE FILE', 99/365,SD

// EXTENT SYS002,111111, 1, 000,00001, 1219

// EXEC CORGZ

ø ALLOC CL=60(10), RL=30(10), SL=30(10)

ø COPY ALL

/*

/&
```

CORGZ	The program phase that performs the copy function.
ALLOC	Same as described in Reallocate function.
COPY ALL	Cause all of libraries to be copied onto the new pack.

EXERCISES

1. Write the job control language necessary to print the directories of the core image library and the source statement library.

2. Write the job control language necessary to cause the following to be printed.

```
0       P.FDMS57     V.M     1.5     2 BLOCKS       SOURCE STATEMENT LIBRARY 0
0                                                                            0
0           BKEND    C.FDMS57                                                0
0                                                                            0
0     00001     RECORD CONTAINS 80 CHARACTERS                                0
0     00002     LABEL RECORDS ARE OMITTED                                    0
0     00003     DATA RECORD IS IT01-CARD-RCD.                                0
0     00004 01  IT01-CARD-RCD.                                               0
0     00005     05 IT01-NAME                     PIC  X(20).                 0
0     00006     05 IT01-SSAN                     PIC  9(9).                  0
0     00007     05 IT01-GRADE                    PIC  X(2).                  0
0     00008     05 IT01-ADDR                     PIC  X(20).                 0
0     00009     05 IT01-PT-SCORE                 PIC  9(3).                  0
0     00010     05 IT01-PHONE                    PIC  9(4).                  0
0     00011     05 IT01-UNIT                     PIC  X(10).                 0
0     00012     05 FILLER                        PIC  X(12).                 0
0                                                                            0
0           BKEND                                                            0
0                                                                            0
```

568

3. Write the job control language necessary to catalog an object module in card deck form into the proper library and call it ZOOM5.

4. Write the job control language necessary to catalog a set of JCL cards into the proper library and call it PROC97.

5. Write the job control language necessary to delete GOPHASE from the core image library and MOD84 from the relocatable library and also condense all three libraries.

654

DOS UTILITY PROGRAMS

1. INTRODUCTION. Whatever the specific uses of a data processing system, certain unique operations exist that must be performed frequently. These operations may differ in detail, depending on the particular machine configuration and data format for the individual user, but the essential function remains the same. Generalized routines designed to satisfy specific functions would ease the burden of programming these operations. These routines must be flexible enough to allow the user to assign the specifications of his particular problem.

One type of program that meets these requirements are utility programs. They are designed to assist the user in day-to-day operation of his installation. With these programs, the user can perform certain frequently required operations, such as transferring disk storage files to cards or tape and printing out areas of tape or disk for program testing purposes, without programming effort.

2. EXPLANATIONS OF THE UTILITY SPECIFICATIONS.

a. File to File Utilities. Eleven utility programs are provided for the transfer of data files from any of the normal input devices to any of the normal output devices. A file can be transferred between unlike storage mediums (tape to disk), like mediums (tape to tape), or, in the case of disk to disk, the files may be transferred from one area to another area of the same unit.

A file can be transferred from one input medium to an output medium with these option:

* COPY - indicates that the file is to be transferred from an input medium without change to the format of the records or file.

* REBLOCK - transfers the input file from an input medium to an output medium with only the block size being changed.

* FIELD SELECT - rearranges or drops fields within each input record, or converts them to zoned or packed decimal.

* REBLOCK AND FIELD SELECT - is a combination of the reblock and field select options.

Printer output allows you to show the output in three ways:

* DISPLAY - allows you to display a byte for byte representation of the information.

* LIST - gives an edited representation of the information.

* LIST AND FIELD SELECT - is a combination of the list and field select options.

For the card to printer and/or punch programs, two other output options are:

- * BOTH PRINT AND PUNCH - is a combination of copy and list.
- * BOTH PRINT AND PUNCH WITH FIELD SELECT - combines copy and list with field select.

These programs will handle fixed-length, variable-length, and undefined-length records, however, only fixed or variable length records can be reblocked or field selected.

b. Label Checking. Tapes containing standard labels or no labels at all can be processed without providing a user routine. When using a file to file utility program to process nonstandard or user labels or when no label checking is desired, an UPSI job control card is required. Only the 0 and 2 bits are normally used. When bit 0 is OFF, standard labels are checked on input and when it is ON, nonstandard or no label input is checked. Bit 2 OFF indicates output standard labels while bit 2 ON indicates nonstandard are not labeled.

When an UPSI card is supplied to a program, the byte is propagated from job step to job step unless another UPSI card is supplied to reset the bits. All of the UPSI bits are set to 0 following each job performed unless a new statement is supplied. When right most bits are not set by an UPSI statement, they are assumed to be zero.

c. Multifile Volume (tape). The utility programs may be used to build multifile volumes and read from them at later dates. File positioning will be performed by logical IOCS only if the files are labeled with standard labels. Output files, nonstandard labeled files and unlabeled files are not acceptable. The file-name, volume-sequence, and file-sequence numbers must be placed in the TLBL card. When using the utility programs to process multifile tape input volumes, the non-rewind-option (IN) parameter found in the utility modifier statement must be specified.

d. Multivolume Files (tape). Input or output files to these programs can consist of multiple volumes and must belong to the same data files. The control statement entries used to process the first volume are used to process each successive volume because the same fields are checked in each volume. Each tape reel of a multivolume tape file is unconditionally rewound and unloaded if no alternate tape drive has been assigned. In all other cases, the volume will be treated as specified by the input or output parameter in the utility modifier statement.

When alternate tape drives are specified and processing is completed on a particular file, the last drive processed will become the primary drive. If a new job is executed at this time, the last drive processed will then become the primary drive unless a reassignment of tape drives is made.

e. Record Skipping. Any number of logical records (up to 99,999,999) may be bypassed before processing is performed. This number can be indicated in the Rx parameter of the utility modifier statement. The number indicated in the parameter will be the first record to be processed. Record skipping cannot be performed for the copy function (TC), and if specified, it will be ignored ignored. To skip records at the beginning of a file and copy the remainder, the reblock function (TR) must be indicated, and the input-description and output-description parameters must contain identical values.

f. Sequence Numbering. Sequence generation on card output can be indicated in the utility modifier statement. A field up to ten characters long can be punched into each card. This field is numbered starting from 1 (with high-order zeros) and is increased by 1 for each succeeding card. If a sufficiently long field is not defined to number all of the cards, the number wraps around to zero without an error indication. The sequence number overlays any data selected into the sequence area of the card. Sequence checking can also be performed for card input to assure ascending sequence of the specified field. If a card is out of sequence, a message is written on SYSLST and processing continues.

g. Binary Records. When processing cards punched in column binary format, the input and output parameters (A and B in the utility modifier statement) must be specified as twice the number of card columns used.

h. Printer Output. Printer output can be 120, 132, or 144 characters per line depending on the printer used and the output format. Printer output can be data display or list format.

* DATA DISPLAY - this format provides a visual picture of the data file. Fixed, variable, and undefined records can be handled, but the field select option cannot be used. Every byte of data in the file appears in the printed output. If data display is specified (TD), 120-character line is forced. Only portions of the print line are used for data. This first twenty (1-20) are reserved for information describing the file, such as block size, block number, and record number. Data is normally displayed in hexadecimal form, but may be optionally displayed in alphabetic form. A heading line can be printed. A scale line is printed at the top and bottom of each page. If record length is specified as fixed or variable length, each logical record starts on a new line. The input block size is printed only if the input length is not equal to the specified block size. The excess is not printed when the maximum length block is exceeded. Single spacing is used between lines of print.

* DATA LIST - the data list format provides a simple edited listing of the file. The entire print line is available for data output; output is restricted to one line per logical record. The input record length cannot exceed the size of the print line. If so, the field select is required. Fields can be selected to unpack, convert to hexadecimal, and format the page. Data list mode allows only character printing unless a hexadecimal field is selected through a field select entry.

i. Utility Modifier Statement. This statement is used to describe the input file that is to be processed and the output file that is desired. If the statement is present and optional parameters are left out, default values are assumed.

The general format of the Utility Modifier Statement is:

```
// Uxx Tt,Ff,A=(input),B=(output),E=(c), Ix,Ox,Px,Q=(x,y),Rx,Sx
```

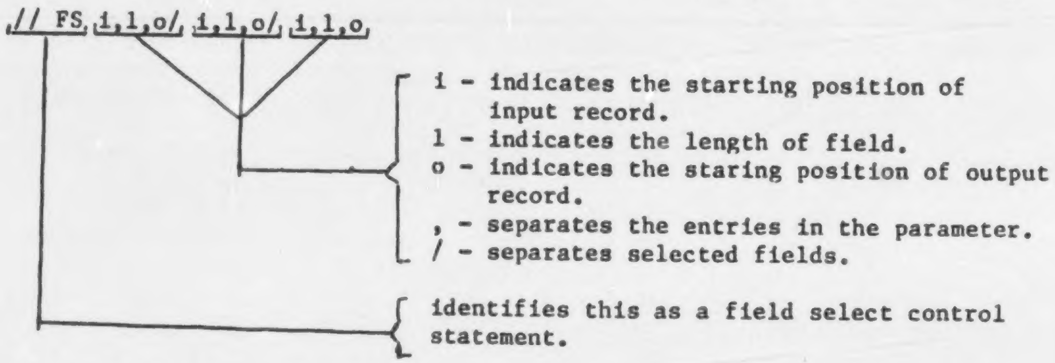
j. Field Select Statement. With this option, a field is each input record can be moved to a different location in the corresponding output record. Those areas of the output record that are not filled with selected fields are filled with blanks.

When field select is used, only those bytes in the input record that are selected will be transferred to the output record. It is therefore possible with field select to ignore certain fields and have them dropped from the output record.

The field select control statement provides the information for the file-to-file programs to transfer fields from an input record to the same or a different relative location of the output record. Each card should begin with //b̸FSb̸. As many field select statements as necessary may be used and must be complete on one statement. For example:

```
// FS 1,15,1/16,(P,5,3),16/72,5,19
// FS 21,2,30
```

The format and contents of the field select statement are:



// FS i,l,o/ i,l,o/ i,l,o

i - indicates the starting position of input record.
l - indicates the length of field.
o - indicates the staring position of output record.
, - separates the entries in the parameter.
/ - separates selected fields.

identifies this as a field select control statement.

k. Heading Statement. A heading line can be printed for programs with printed output. Heading lines are ignored if first-character forms control is specified. A maximum of two statements can be used to indicate the heading line desired, but the second statement need not be entered if the first statement contains all of the desired information. The first statement is entered // H1b̸ (followed by the information to be printed in print positions 1-74. The second heading statement is // H2b̸ (followed by the remainder of the heading line). Heading statements must follow any field select statements used.

l. End Statement. This statement must be the last of the utility control statements in the program and is punched // END.

3. FILE TO FILE UTILITY PROGRAMS.

a. Card to Disk. The card to disk program transfers the contents of a card file from cards to an area of disk. The cards may be punched in EBCDIC or binary. The input records must be fixed-length unblocked, and each logical record must fit on one card. The maximum size input record is 80-bytes or 160-bytes for binary. These files may be copied, reblocked, field selected, or reblocked and field selected.

b. Card to Printer and/or Punch. Input records to this program must be fixed-length and unblocked. Card input and output can be either EBCDIC or binary except for both, printing and punching when it must be EBCDIC. The maximum record size is 80-bytes for EBCDIC and 160-bytes for binary.

(1) Card to Printer. The card to printer program can produce printed output in two formats: Display and List. Sequence checking is performed on the input.

(a) Display. This option transfers to contents to a card file to a printer with each record being placed on one print line. The field select option cannot be performed with display. In this format, the first 20 positions of the print line are reserved for information describing the file. When hexadecimal printout is called for, the entire card is printed on two lines.

(b) List. The input records for this option are transferred to the printer with each record being fully printed. The field select option may be used. The full print line is available for printing. When hexadecimal printout is called for, the output record size is bound by the size of the print line.

(2) Card to Punch. Records may be copied or field selected. Sequence fields are generated, but input is not checked.

(3) Card to Printer and Punch. This program produces printed output in the list format and a reproduced card deck. Sequence fields are generated, but input is not checked.

c. Card to Tape. The card to tape program transfers the contents of a card file to tape. The cards may be punched in EBCDIC or binary. The input records must be fixed-length unblocked and each logical record must fit on one card. The maximum record size is 80-bytes for EBCDIC or 160-bytes for binary.

d. Disk to Card. The disk to card program transfers the contents of a disk file to a card file. The output file may be punched in either EBCDIC or binary. Each logical output record must fit on one card. Input records to this program must be fixed-length and may be blocked. Files in this program may be copied, reblocked, field selected, or reblocked and field selected.

574
All blocked input records must be reblocked.

e. Disk to Disk. The disk to disk program transfers a file between disk units or between areas of the same unit. Using the same device for input and output can cause a reduction in performance. Files can be copied, reblocked, field selected, or reblocked and field selected. If the field select or reblock options are used, the input records must be fixed or variable length.

f. Disk to Printer. The disk to printer program can display a disk file in either data display or data list format. Data display provides a visual picture of the data where every byte appears in the printed output; this format can handle fixed, variable, and undefined records. Data list provides a simple edited list of the file; input records must be fixed or variable length and (for the list function only) must not exceed the size of the print line. An option is available to this program to specify the number of logical records in a file to be bypassed before printing begins.

g. Disk to Tape. The disk to tape program transfers a file from one or more disk units to one or more tape reels. These files may be copied, reblocked, field selected, or reblocked and field selected. If the reblock or field select options are used, the input records must be fixed or variable length.

h. Tape to Card. This program transfers a tape file to a card file. The output file may be punched in either EBCDIC or binary. Input records must be fixed length. The files may be copied, reblocked, field selected, or reblocked and field selected. Blocked input records must be reblocked.

i. Tape to Disk. The tape to disk program transfers a file from one or more tape reels to a maximum of 'N' disk units where 'N' is the number of disk units assigned. These files may be copied, field selected, reblocked, or reblocked and field selected. If the field select or reblock options are used, the input records must be fixed or variable length.

j. Tape to Printer. The tape to printer program can display a tape file in either data display or data list format. Data display provides a byte-for-byte representation of the data file where every byte appears in the listing; this format can handle fixed, variable and undefined records. Data list provides a simple edited representation of the file; input records must be fixed or variable length and (for the data list function only) must not exceed the size of the print line. The field select option may be used. An option is available to this program to specify the number of logical records in a file to be bypassed before printing begins.

k. Tape to Tape. The tape to tape program transfers a file from one or more reels to tape to one or more other reels. These files may be copied, field selected, reblocked, or reblocked and field selected. Input records must be fixed or variable length if the reblock or field select options are used.

---

4. PHASE NAMES FOR UTILITY PROGRAMS. The phase names listed below are for the utility programs described above.

PHASE NAME	PROGRAM DESCRIPTION
CDDK	Card to disk
CDPP	Card to printer and/or punch
CDTP	Card to tape
DKCD	Disk to card
DKDK	Disk to disk
DKPR	Disk to printer
DKTP	Disk to tape
TPCD	Tape to card
TPDK	Tape to disk
TPPR	Tape to printer
TPTP	Tape to tape

5. UTILITY MODIFIER STATEMENT PARAMETER. The contents of the utility mofifier statement parameter is described below.

FUNCTION

TB - Both print and punch
TBF - Both print and punch with field select
TC - Copy
TD - Display
TF - Field Select
TL - List
TLF - List and field select
TR - Reblock
TRF - Reblock and field select

FORMAT

FF - Fixed length
FV - Variable length
FU - Undefined length

INPUT DESCRIPTION

A=(n,m) where 'n' is record length and 'm' is block length

OUTPUT DESCRIPTION

B=(n,m) where 'n' is record length and 'm' is block length
B=(p) where 'p' is normally 132 for a 1403N1 printer

INPUT MODE

I1 - EBCDIC
I2 - Binary
IN - Do not rewind before or after data transfer (tape only)

INPUT MODE - continued

IR - Rewind before and after data transfer (tape only)
IU - Rewind before and after and unload (tape only)

OUTPUT MODE

O1 - EBCDIC (punch only)
O2 - Binary (punch only)
ON - Do not write disk check (disk only) or do not rewind before or after data transfer (tape only)
OC - Character (printer only)
OR - Rewind before and after data transfer (tape only)
OU - Rewind before and rewind and unload after data transfer (tape only)
OX - Hexadecimal (printer only)
OY - Write disk check (disk only)

SEQUENCE NUMBERING

Q=(x,y) where 'x' is first position of a field for sequence numbering (1 or 2 digits) and 'y' is length of field not to exceed 10 positions

PAGE NUMBERING

PN - Do not nomber pages
PY - Number pages

FIRST RECORD

Rx - where 'x' is the first record to be processed, normally 1

SPACING

S1 - Single space or stacker select pocket 1
S2 - Double space or stacker select pocket 2
S3 - Triple space or first character stacker control
SA - These
SB - codes are
SC - not normally
SC - used

6. COMMONLY USED UTILITY MODIFIER STATEMENT PARAMETERS. The list shown below are the most frequently used utility modifier parameters.

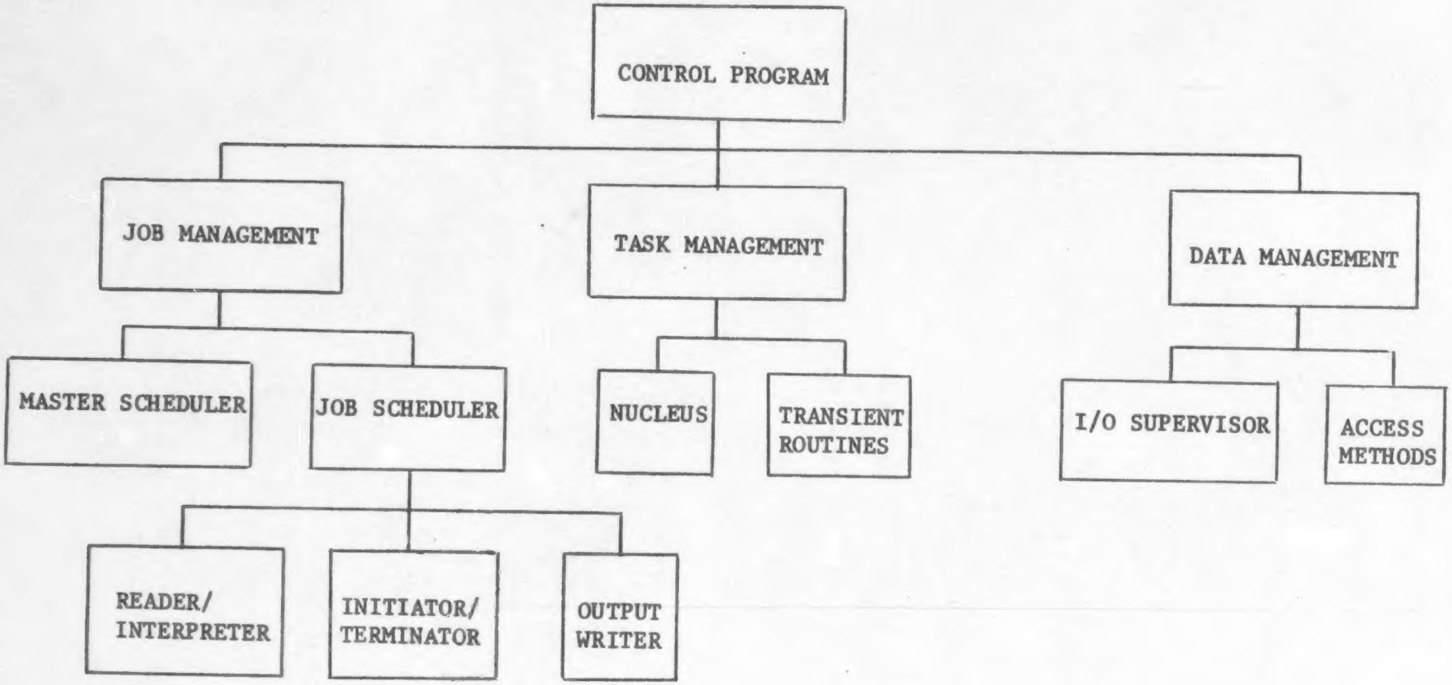
```
// EXEC CDDK   (card to disk)
// UCD TR,FF,A=(80,80,B=(80,1680),E=(2314),R1,OY

// EXEC CDPP   (card to card)
// UCP TC,FF,A=(80,80),B=(80,80),I1,O1,R1,S1
```

COMMONLY USED UTILITY MODIFIER STATEMENT PARAMETERS - continued

```
// EXEC CDPP    (card to print double space)
// UCP TL,FF,A=(80,80),B=(132),I1,OC,PY,S2,R1

// EXEC CDPP    (card to card and print single space)
// UCP TB,FF,A=(80,80),B=(80,80),I1,OC,PY,R1,S1

// EXEC CDTP    (card to tape)
// UCT TR,FF,A=(80,80),B=(80,1600),I1,OR,R1

// EXEC DKCD    (disk to card)
// UDC TR,FF,A=(80,1680),B=(80,80),R1,S1,E=(2314)

// EXEC DKDK    (disk to disk)
// UDD TR,FF,A=(80,1680),B=(80,1680),E=(2314,2314),OY,R1

// EXEC DKPR    (disk to print double space)
// UDP TL,FF,A=(80,1680),B=(132),E=(2314),PY,OC,S2,R1

// EXEC DKTP    (disk to tape)
// UDT TR,FF,A=(80,1680),B=(80,1600),E=(2314),R1,OR

// EXEC TPCD    (tape to card)
// UTC TR,FF,A=(80,1600),B=(80,80),IU,01,R1,S1

// EXEC TPDK    (tape to disk)
// UTD TR,FF,A=(80,1600),B=(80,1680),E=(2314),IR,OY,R1

// EXEC TPPR    (tape to print double space)
// UTP TL,FF,A=(80,1600),B=(132),IU,PY,R1,S2

// EXEC TPTP    (tape to tape with field select)
// UTT TRF,FF,A=(80,80),B=(81,81),IU,OU,R1
```

7. For additional information on the utility programs, refer to IBM SRL GC 24-3465, DOS and TOS Utility Programs.

THE OPERATING SYSTEM (OS)


CONTROL PROGRAM
JOB MANAGEMENT
TASK MANAGEMENT
DATA MANAGEMENT
MASTER SCHEDULER
JOB SCHEDULER
NUCLEUS
TRANSIENT ROUTINES
I/O SUPERVISOR
ACCESS METHODS
READER/ INTERPRETER
INITIATOR/ TERMINATOR
OUTPUT WRITER

Job control language for COBOL PE 2 (OS)

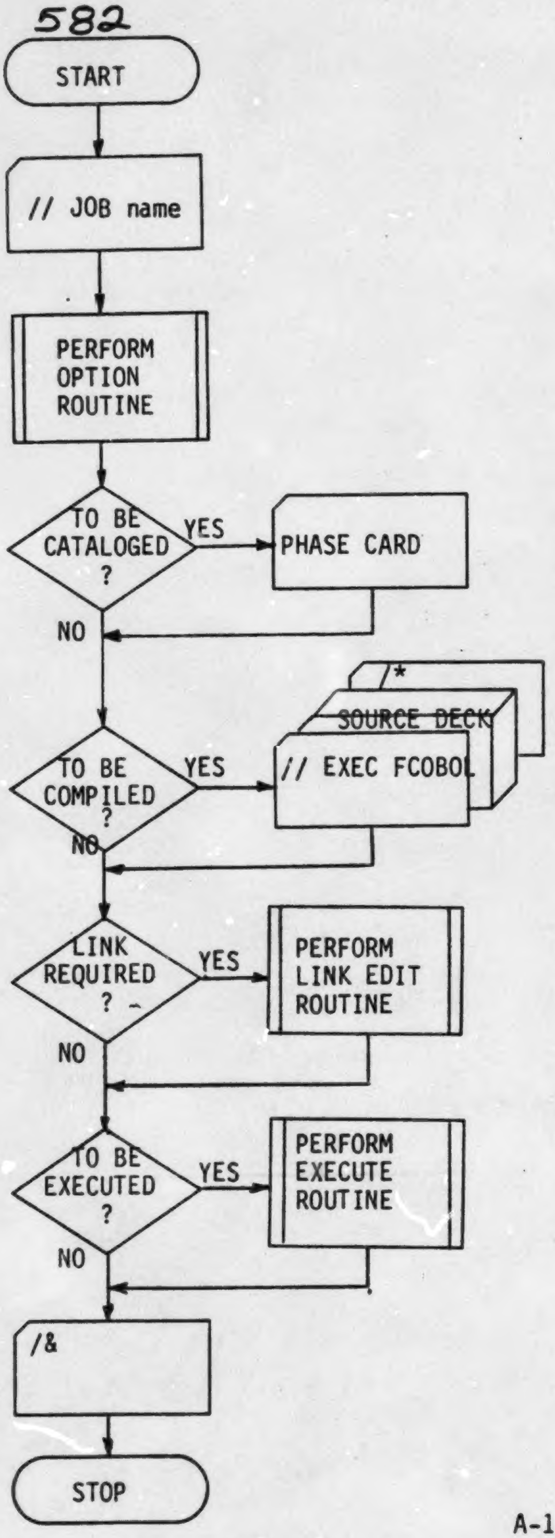
```
//C23        PROC DUMP='DUMMY'
//*    THIS PE READS NAMES AND LISTS THEM
//*    CREATED ON 30 NOV 76 BY SOFTWARE DIV, CSD, USAIA
//COB        EXEC PGM=IKFCBL00
//           PARM='LOAD,SUPMAP,SIZE=84K,BUF=8K,NODECK,CLIST'
//STEPLIB    DD DSN=USAIA.SOFTWARE.LINKLIB,DISP=SHR
//SYSPRINT   DD SYSOUT=A
//SYSPUNCH   DD DUMMY
//SYSLIB     DD DSN=USAIA.SOFTWARE.SOURCE,DISP=SHR
//SYSUT1     DD UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT2     DD UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT3     DD UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT4     DD UNIT=SYSDA,SPACE=(460,(700,100))
//SYSLIN     DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//              SPACE=(80(500,100))
---------------------- COBOL source deck would be placed here ---------------
//LKED       EXEC PGM=IEWL,PARM='LIST,XREF,MAP,LET',COND=(8,LT,COB)
//STEPLIB    DD DSN=USAIA.SOFTWARE.LINKLIB,DISP=SHR
//SYSLIN     DD DSN=&LOADSET,DISP=(OLD,DELETE)
//SYSLMOD    DD DSN=&GOSET(COB),DISP=(NEW,PASS),UNIT=SYSDA,
//              SPACE=(1024,(50,20,1))
//SYSLIB     DD DSN=SYS1.COBLIB,DISP=SHR
//SYSUTI     DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))
```

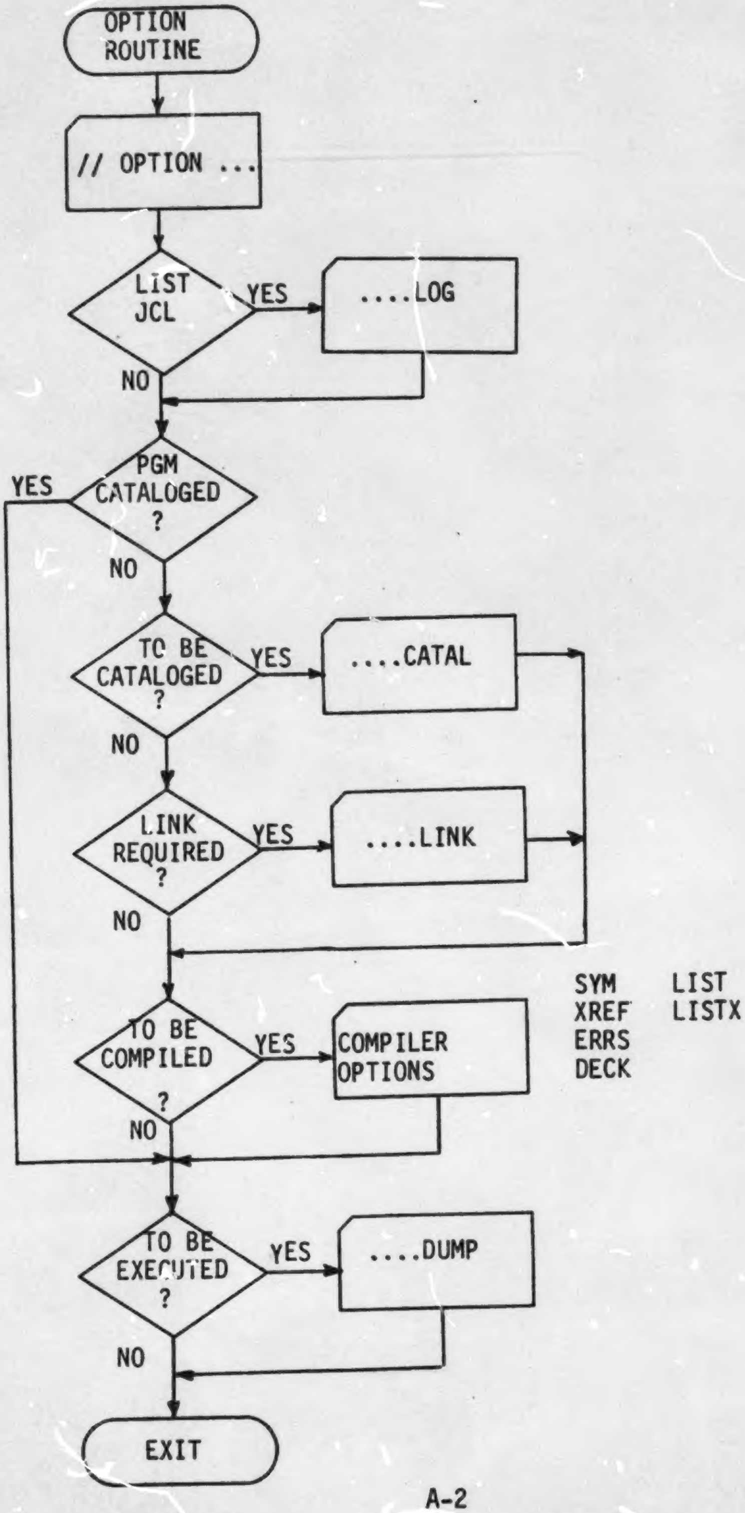
```
//SYSPRINT  DD SYSOUT=A
//GO        EXEC PGM=*.LKED.SYSLMOD,COND=((8,LT,COB),(4,LT,LKED))
//SYSOUT    DD SYSOUT=A
//SYS005    DD SYSOUT=A
//SYS006    DD DSN=USAIA.SOFTWARE.DATASET.C09,DISP=SHR
```

OS JCL to execute a COBOL PE

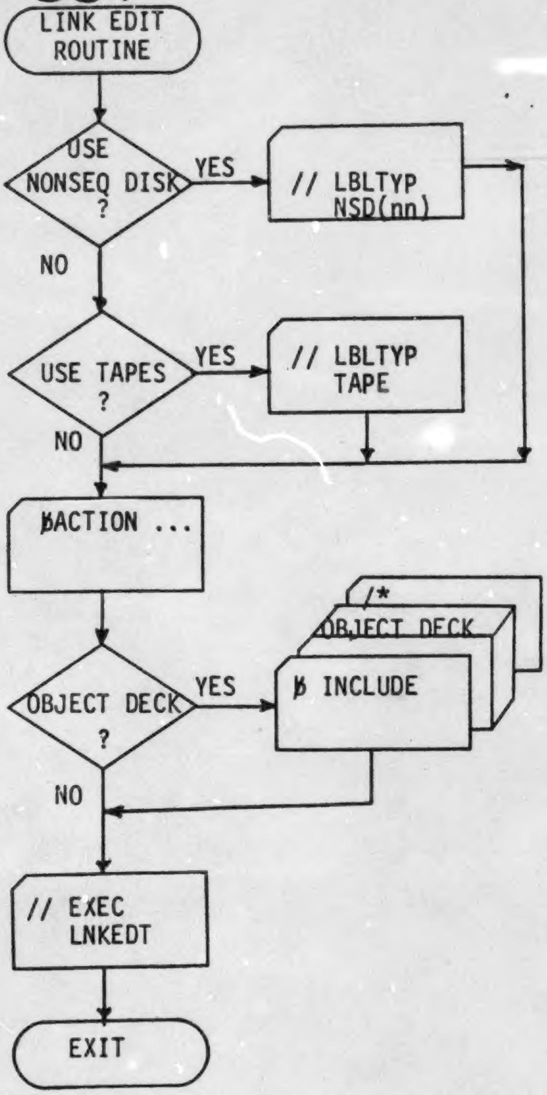
```
//name          JOB (procname,room),'programername',MSGLEVEL=(1,1)

//              EXEC procname

//COB.SYSIN     DD  *

                  COBOL
                  source
                  deck

/*

//
```

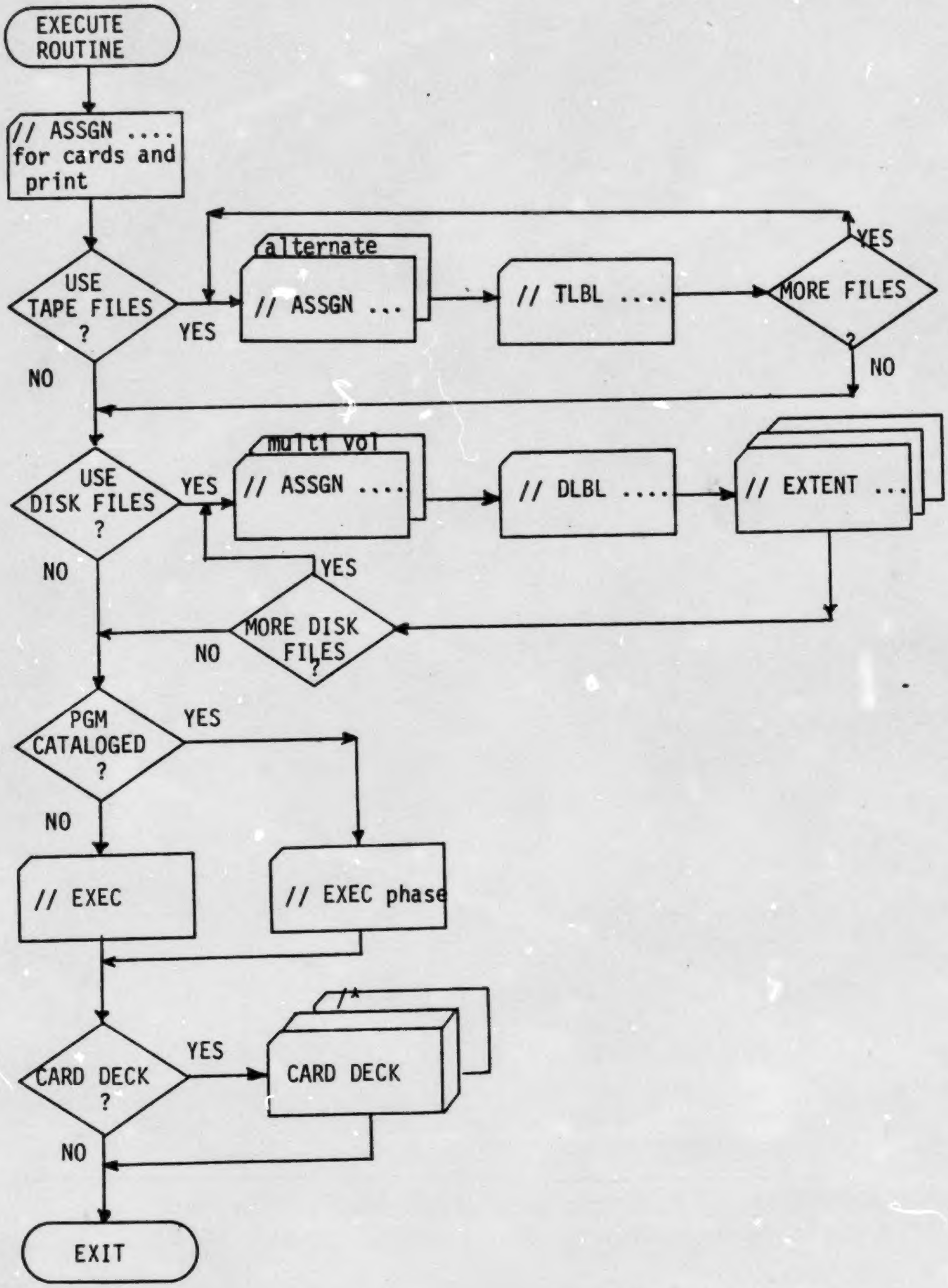
name	This is a name assigned to the job. This parameter is taken from the 1st eight characters of your name on the 'buff colored' control card you now put on your deck.
procname	Use the code from the buff colored card C11,C09,C23,....
room	Enter your classroom number
programername	'John Q. Smith'
procname	Same as above.


START
// JOB name
PERFORM
OPTION
ROUTINE
TO BE
CATALOGED
?
YES
PHASE CARD
NO
/*
SOURCE DECK
TO BE
COMPILED
?
YES
// EXEC FCOBOL
NO
LINK
REQUIRED
?
YES
PERFORM
LINK EDIT
ROUTINE
NO
TO BE
EXECUTED
?
YES
PERFORM
EXECUTE
ROUTINE
NO
/&
STOP

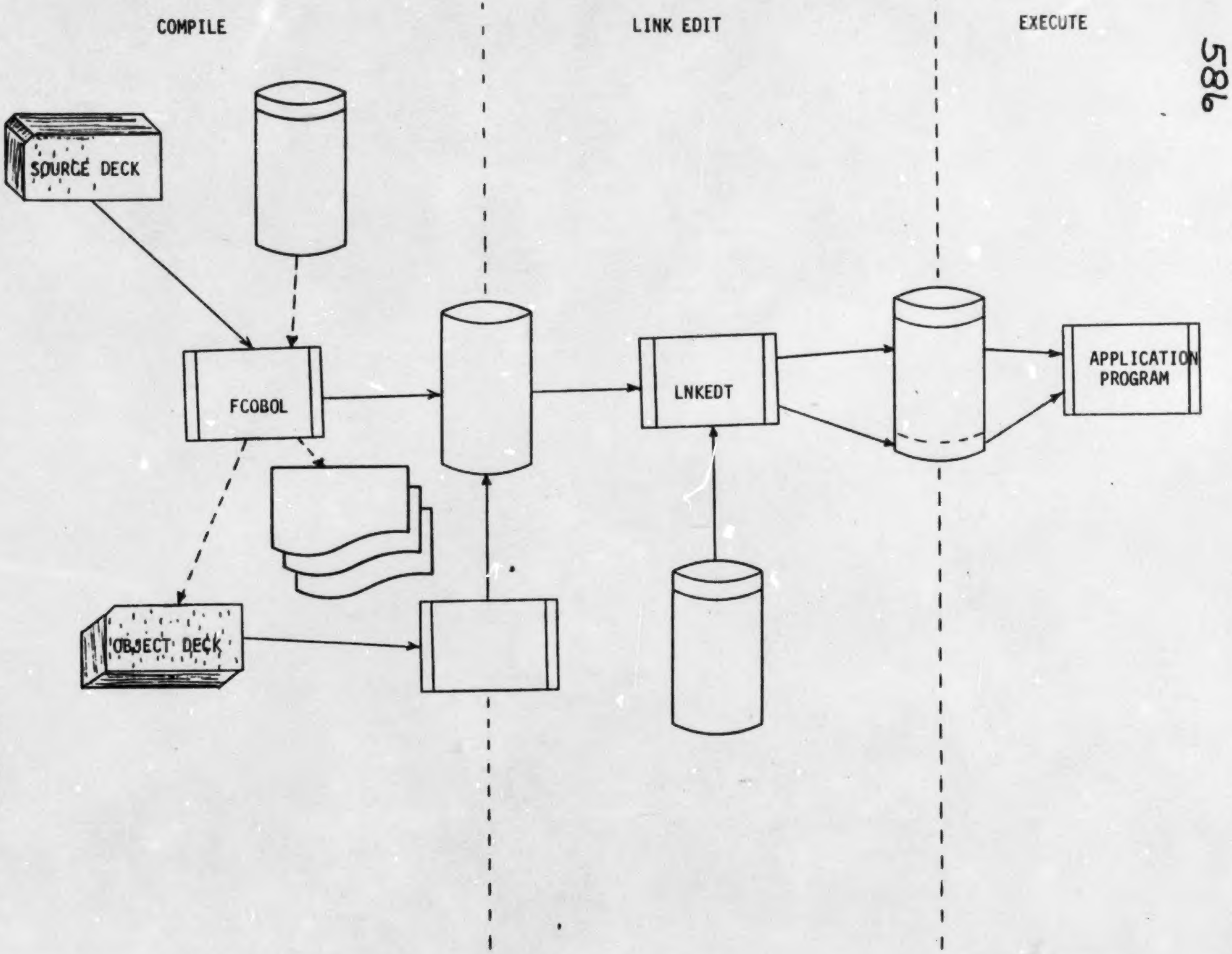

OPTION ROUTINE
// OPTION ...
LIST JCL
YES
....LOG
NO
PGM CATALOGED ?
YES
NO
TO BE CATALOGED ?
YES
....CATAL
NO
LINK REQUIRED ?
YES
....LINK
NO
TO BE COMPILED ?
YES
COMPILER OPTIONS
SYM
XREF
ERRS
DECK
LIST
LISTX
NO
TO BE EXECUTED ?
YES
....DUMP
NO
EXIT

584


LINK EDIT ROUTINE
USE NONSEQ DISK ?
YES
// LBLTYP NSD(nn)
NO
USE TAPES ?
YES
// LBLTYP TAPE
NO
ƀACTION ...
OBJECT DECK ?
YES
/*
OBJECT DECK
ƀ INCLUDE
NO
// EXEC LNKEDT
EXIT


EXECUTE ROUTINE
// ASSGN .... for cards and print
USE TAPE FILES ?
YES
NO
alternate
// ASSGN ...
// TLBL ....
MORE FILES ?
YES
NO
USE DISK FILES ?
YES
NO
multi vol
// ASSGN ....
// DLBL ....
// EXTENT ...
MORE DISK FILES ?
YES
NO
PGM CATALOGED ?
YES
NO
// EXEC
// EXEC phase
CARD DECK ?
YES
NO
/*
CARD DECK
EXIT

586


COMPILE
LINK EDIT
EXECUTE
SOURCE DECK
FCOBOL
OBJECT DECK
LNKEDT
APPLICATION PROGRAM

673

674

IA-01-01-05 (D)

SYS/360 OPERATING CONCEPTS

OBJECTIVES:

1. To describe in detail the three mainframe components of the IBM SYS/360 computer, Main Memory, the Central Processing Unit, and Data Channels.
2. To describe the five machine language formats.
3. To describe interrupts and interrupt processing.
4. To define PSW, IOCS, LIOCS, and PIOCS.

I. MAIN MEMORY - Where data and instructions must ultimately be to be operated upon by the system.

  A. Byte -

  B. Addressing -

  C. Concept of "Word Units"

    1. Fullword -

    2. Halfword -

    3. Doubleword -

    4. Boundary Alignment

II. CENTRAL PROCESSING UNIT (CPU).

  A. Instruction Cycle.

    1. I-Time -

    2. D-Time -

    3. The Difference Between Instructions and Data -

B. Access Width -

C. Access Time -

D. Control Element - Fetches and decode machine instructions and sets up internal logic paths for the actual execution of the instruction.

E. Arithmetic and Logical Unit (ALU) - Responsible for actually executing the instructions as decoded by the Control Element.

1. General Purpose Registers (GPR).

   a. Size and Number -

   b. Data Format -

   c. Uses

      1)

      2)

2. Floating Point Registers (FP REG).

   a. Size and Number -

   b. Data Format -

   c. Use

3. Operations.

   a. Arithmetic -

   b. Logical -

III. DATA CHANNELS - Special purpose minicomputers whose only function is to monitor I/O operations.

A. Modes of Operation.

1. Burst -

2. Multiplex -

B. Types of Channels.

1. Selector -

2. Multiplexor -

C. Channel Command Word (CCW).

D. Characteristics of Channels.

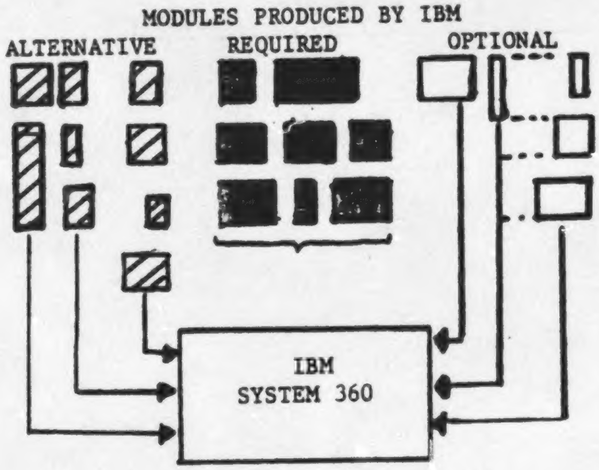
1.

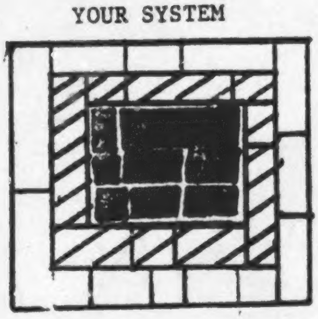
2.

3.

590

IV. THE MODULAR CONCEPT OF THE SYS/360.


MODULES PRODUCED BY IBM
ALTERNATIVE
REQUIRED
OPTIONAL
IBM
SYSTEM 360


YOUR SYSTEM

V. MACHINE LANGUAGE INSTRUCTION FORMATS.

A. The Two Theories of Machine Language Instruction Formats.

1. Fixed Length
   a.
   b.
   c.
2. Variable Length
   a.
   b.
   c.

678

B. Parts of a Machine Language Instruction.

1. Operation Code (OPCODE) - The first byte of the instruction is always the OPCODE. It defines the arithmetic or logical or I/O operation that is to be done.

2. Operands.

   a. Registers -

   b. Storage -

   c. Data -

C. Formats.

1. Register to Register (RR).

   a. Length -

   b. Operands -

   c. Diagram -

   c. Type Instructions -

2. Register to Storage.(RS).

   a. Length -

   b. Operands -

   c. Diagram -

   d. Type Instructions -

3. Register to Indexed Storage (RX).

   a. Length -

   b. Operands -

c. Diagram

d. Type instructions -

4. Storage and Immediate (SI).

   a. Length -

   b. Operands -

   c. Diagram -

   d. Type Instructions -

5. Storage to Storage (SS)-

   a. Length -

   b. Operands -

   c. Diagrams -

   d. Type Instructions -

VI. Interrupts - The mechanism that allows the machine to change from the problem program stat to the supervisor state.

A. Types of Interrupts.

1.

2.

3.

4.

5.

B. Priority of Interrupts for Simultaneous Occurances.

   1. Stacking Priority.

      a.

      b.

      c.

      d.

   2. Execution Priority.

      a.

      b.

      c.

      d.

   3. Handling of Simultaneous Interrupts

C. Program Status Word (PSW).

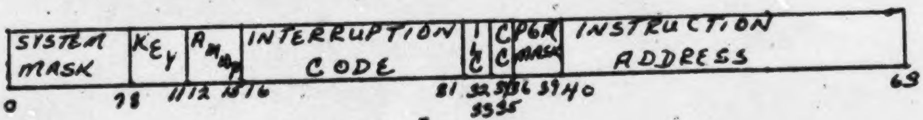
   1.

   2. Current PSW -

   3. New PSW -

   4. Old PSW -

   5. Diagram -


SYSTEM MASK
KEY
AMWP
INTERRUPTION CODE
IL C
CC
PGM MASK
INSTRUCTION ADDRESS
0
78
1112
1516
31
32 33
34 35
36 39 40
63

594

D. Interrupt Processing.

1.

2.

3.

4.

5

VII. Sys/360 Input/Output

A. IOCS. IOCS stands for Input Output Control System. In the supervisor there is a small block of coding which serves to initiate I/O operations in response to program requests. It also notifies your program that an I/O operation is complete.

B. PIOCS. Physical IOCS is the most basic method of accessing the IOCS section of the supervisor. In PIOCS the programer has direct control over I/O operations but is responsible for all error checking and program notification of completed I/O.

C. LIOCS. Logical IOCS is the system of accessing I/O devices most often used. LIOCS provides for all error handling and program notification of completed I/O. In order to do this there are 3 parts to the LIOCS system:

1. A table used to define the appearance of the file being used. This table is called a Define The File table or DTF.

2. A block of executable coding to handle error checking, I/O initiation, etc., called a module.

3. An I/O imperative specifying reading or writing on the file.

Keywords: TABULAR-MODULAR STRUCTURE, I/O AREA, DEVICE ADDRESS, MODULE NAME, PROGRAMER OPTIONS

In order for the 360 to perform an I/O operation three things must be done:

1. Specific information relating to the file must be made available in a table called a DTF table.

2. Coding (a piece of program) must be available to reference the table and pass the information to the channel. This block of coding is called an I/O module.

3. A specific command must be given to initiate the process.

The following information is needed in the table:

1. A Device Address consisting of a logical unit address must be provided. This address is then used to locate the physical address of the unit.

2. The location of the data must be specified explicitly. This area is called an I/O Area and is first defined by the programer. This area is used to hold data for output or if it is an input area to receive data. Usually there is at least one I/O area per file.

3. Any special device dependent options are also specified. These are called programer options. An example would be how a tape is blocked or whether a tape is to be rewound at the end of the job.

4. The assembler takes into account all of the above information and determines which relocatable module will do the job.

I/O Module

The I/O module is resident in the relocatable library. There are I/O modules for each set of options for each device.

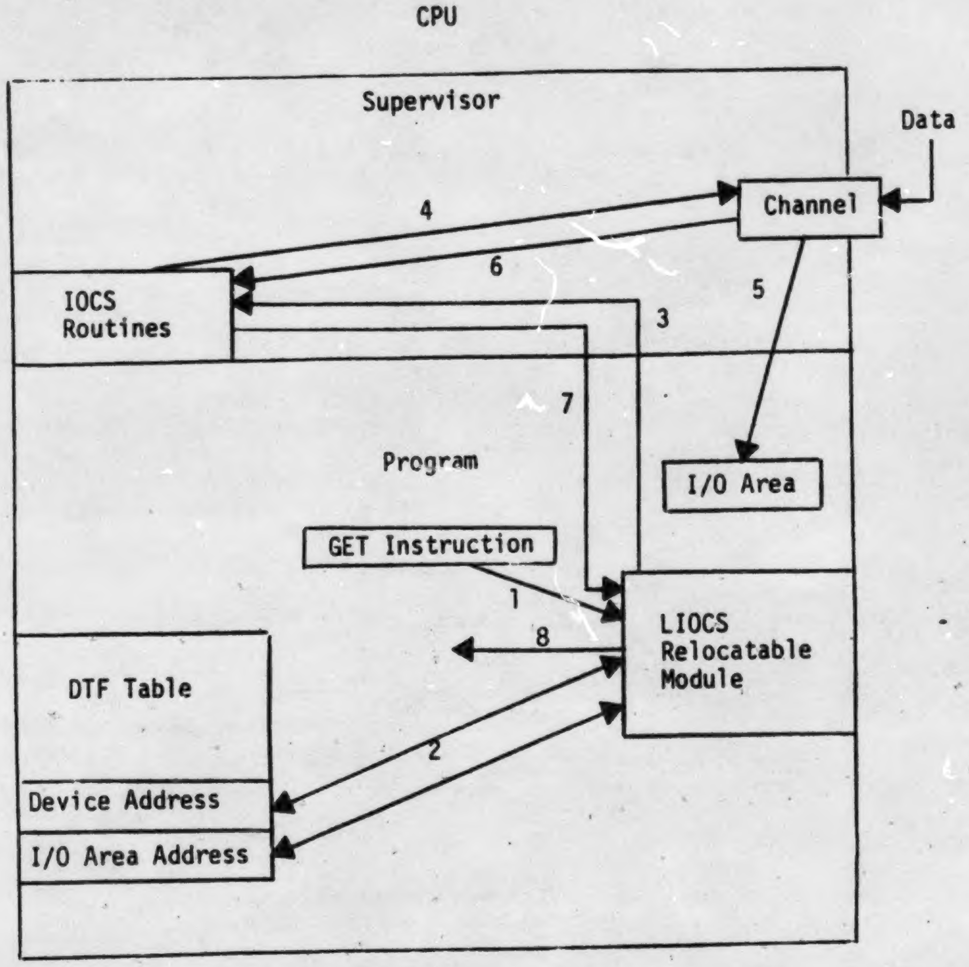
The linkage editor is given the name of the appropriate module in the external address portion of the symbol table. It then pulls it off the relocatable library and links it in to your program.

The I/O module provides an interface between your program and the IOCS routines in the supervisor. It also provides for error checking and program notification of completed I/O. It also provides the coding necessary for implementation of programer options.

596

Imperative Command

In order to access the table and module for I/O the programer need only use the GET (filename) or PUT (filename) commands. These commands provide branches to the appropriate I/O routines.


CPU
Supervisor
Data
Channel
4
6
5
3
IOCS Routines
7
Program
I/O Area
GET Instruction
1
8
LIOCS Relocatable Module
DTF Table
2
Device Address
I/O Area Address

1. GET instruction branches to module.
2. Module obtains information it needs from DTF table.
3. Module passes information to IOCS routines in supervisor and requests I/O initiation.
4. IOCS routine passes information to channel and starts I/O.
5. Channel places data in I/O area in core.
6. Channel notifies IOCS of I/O completion.
7. IOCS passes completion information to module. The module then analyzes completion code to determine what error conditions, if any, are present.
8. If no errors, control is passed back to program.

597

Computer Systems Command Interface With the Field Data Processing Activity

A Self-Paced Referenced Text

Programmer/Analyst Course
Software Division, Computer Science Department

January, 1979

IA-01-04-26

INTRODUCTION

OBJECTIVE: Given a situation at a field DPA running standard Army systems, be able to report an incident or a recommended system change on the proper forms and specifying the required information.

CRITERION TEST: The test for this class is different from the others in the course in several ways. First of all, the test is on a go/no go basis, that is, you either get all the points or you get none of them. Secondly, the test is designed to have you demonstrate your ability to perform the two primary CSC interface tasks. You will be given a copy of the test at the beginning (not the end) of the class time allotted to this block. Finally, the test will be graded with you present and, if it is not satisfactory, you will be told what is incorrect and asked to do the test over. This is because we believe that a test should be a learning tool as well as a demonstration of your performance.

METHOD OF LEARNING: This is a self-paced module and as such, you determine how fast you want to learn. If you have any questions on this topic, you may talk with other students or with the instructor. When you have finished the body of this text, complete the criterion test and take it to the instructor for one-to-one feedback. The only restrictions are the test must be finished by the end of the class time allotted and you may not talk with other students when taking the exam.

PURPOSE

This text was designed to acquaint you with how the Army maintains it's standard software systems and the part that you, the programmer, play in the overall scheme. Speaking in more specific terms, this class will show you how to report problems with systems that were not written by you but are running at your DPA. By the way, if you do not know what that abbreviation means, there is a glossary in the back of this text that was designed for people who are not familiar with the Army's unique ADP terminology. Chances are that if you find other unfamiliar terms during this class you will find them defined in the glossary also.

WHY SOFTWARE STANDARDIZATION?

Although the Army has many different units scattered throughout the world, their data processing requirements are really quite similar. They all have, for instance, the following needs:

- Keep personnel data current and available.
- Requisition supplies.
- Keep soldier's pay data current.
- Maintain accountability of government property.

If each Army DPA had to develop software systems to accomplish these and other tasks, there would have to dozens of programmers at the DPA and lots of spare run time to compile and test programs. Additionally, people moving from Post to Post (a fact of Army life) would have to spend many hours being retrained on new systems that did basically the same tasks as the systems at their last Post. With a lack of standardization among ADP systems, attempting to send data between these systems would be a nightmare.

CSC COMES INTO THE PICTURE

To alleviate these problems, the Army designated the Computer Systems Command (CSC) as the central design agency for all ADP systems that are run at more than one command. Under direction of the functional proponent (it's in the glossary), CSC develops standard software systems and sends them to all DPA who have a requirement for them. Not all DPA run the same standard systems, (for instance: a DPA at an Infantry Division would not have the same ADP requirements as the DPA that supports Fort Harrison). When CSC sends a system out it comes in a package containing operator's documentation, JCL, executable phases and instructions on how to load the phases on the libraries. Notice that there are no source programs or programmer documentation in the package. This is to preclude individual commands from changing the programs and therefore not having the same system as other commands. Some of the systems developed by CSC that you are likely to encounter are:

- SIDPERS - Maintains complete personnel data and prints personnel reports. Used at all Divisions and Posts.
- SAILS - Standard supply system at Posts, Corp and supply inventory and management centers.
- STANFINS - The system that keeps track of Army spending at each Post.
- DLOGS - Used at all Divisions for requesting supplies and maintaining property records.

These are just some of the systems that CSC is responsible for. They actually have many systems varying:

- in complexity from under 10 programs to over 100.
- in Hardware from a UNIVAC 1005 to an IBM 370.
- in scope from commissary management to Army Force Accounting.

WHAT DO I DO?

At this point you may be asking yourself what could you possibly have to do with all this since all CSC systems have already been developed and made operational. The answer to this question is why you are having this class!

When you go to a permanent assignment at a DPA you will probably be responsible for assuring the correct operation of an AMIS (remember the glossary!). Your duties in this area will be to:

- Load system changes to your libraries to keep the system current.
- Troubleshoot problems occurring during the execution of an AMIS, and report problems that cannot be overcome to the Assigned Responsible Agency (ARA).
- Forward recommended changes of an AMIS to the ARA.

We will cover each of these tasks separately in the following paragraphs.

KEEPING THE SYSTEM CURRENT

Just as the Army requirements and standards change from time to time so do the AMIS that support the Army. CSC produces system change packages (SCP) on a regular basis, usually every 3 - 6 months, that will keep the system current with Army policy. An SCP comes with all necessary documentation and load instructions. All that is required of the programmer (you) is to forward the package to the operators with any local run instructions and review the output from the SCP load to make sure that the system was loaded correctly. Following the load, you must notify CSC of the date and time of the load. You should be sure to load the SCP as soon as possible so that your system is the current one.

Occasionally, there will be emergency requirements to change a system, usually due to a program failure. When this occurs, CSC sends an Emergency/Urgent Change Package (EUCP) to all affected DPA. EUCP are normally not mailed but are sent through the Automatic Digital Network (AUTODIN) and will arrive within 2 days of dispatch. These change packages must be loaded immediately upon receipt.

WHAT IF THE SYSTEM DOESN'T WORK RIGHT

If an AMIS does not work right, either from an ADP standpoint or a functional standpoint, this means that an incident has occurred. Analyze the incident to make sure it is not caused by a local problem. Once you determine that the problem is not local, report it to CSC as soon as possible by phoning the Customer Assistance Office (CAO). The information that the CAO needs can be taken directly from the Incident Report form, (USACSC Form 53) that you must fill out for your records. Let's take a look at this form in detail, line by line. Refer to the sample form while you read the items below.

1. a. Enter your post name.

   b. This is a number consisting of three fields:

      - The AMIS code

      - Your DPA code

      - A three position sequence number
        a sample number would be: C02-R111-027

c & d. This code designates your DPA type and will be furnished you at your DPA.

e & f. Your name and phone.

g. The hours (in local time) when you work.

h. Your name (unless someone else phones the incident to CSC).

2. a. Leave blank.

b. AMIS Name.

c & d. Self-evident.

e. The person you talked to at CSC.

f. CAO phone number.

3. a. If the incident caused the job to blow-up indicate that and enter the termination code printed on the output listing.

b. Self-evident.

4. This is the number of the SCP or EUCP that was loaded most recently and the date it was loaded (installed). If the problem involves a program, enter the program name in block A or if the problem is in the system documentation enter the documentation name in block B.

5. a. Enter the job name here.

b. Explain in detail exactly what happened. This is very important because it is from this description that CSC will start to resolve the problem.

c. Explain what you did, if anything, to get around the problem.

d. What effect does this incident have on you?

e. Leave blank.

6. Most often this will be applications software.

7. Which of these items are available for CSC to refer to, if needed.

You should maintain a file of all incident reports and indicate in that file the action taken by CSC to resolve the problem.

603

INCIDENT REPORT

For use of this form, see USACSC REG 18-21-1; the proponent agency is Quality Assurance Directorate, (Test & Conf Mgt Div)

1A. INSTALLATION Fort Harrison - USAIA
B. ORIGINATOR NUMBER P02-R111-006
C. ORIGINATOR LEVEL: I
D. OPERATING ENVIRONMENT: 0 1
E. CONTACT SP4 Smith
F. CONTACT'S PHONE 699-2323

2A. CAO ID ______
B. FUNCTIONAL AMIS SIDPERS
C. DATE RECEIVED 12 Jan 79
D. TIME RECEIVED 1600
E. RECEIVED BY SFC Jones
F. PHONE 354-1616

G. CONTACT WILL BE AVAILABLE 0730-1630
H. PERSON PHONING IN INCIDENT SP4 Smith PHONE ______

3. OPERATING STATUS AT TIME OF INCIDENT
A. WAS THERE A CYCLE HALT OR ABNORMAL TERMINATION IN THE PRODUCTION RUN:
☒ YES ABEND CODE WRONG LENGTH RECORD
☐ NO
B. DATE/TIME INCIDENT DETECTED:
1200 Jan 79, 1500

4. BASELINE P02-21-05 INSTL'D 5 Jan 79
A. PROB PGM ID P3AAAC VERS ______
OR
B. PROB DOC ID ______
CHG LEVEL ______ PARA(S) ______
______ PAGES ______

5. DESCRIPTION OF INCIDENT.
A. ID OF RUN IN PROGRESS DAILY CYCLE
B. NARRATIVE DESCRIPTION OF INCIDENT AND RELATED EVENTS Job aborted due to wrong length record on input tape A3AAAC. Listed tape and saw that all records were one byte too long.

C. ACTION TAKEN BY USER AND RESULTS
Used TPTP utility to strip the extra byte and continued.

D. IMPACT ON USER UNTIL RESOLVED
Every time the cycle is run, the tape must be reformatted.

E. CAO/ASD REDEFINITION OF PROBLEM IF DIFFERENT THAN 5B.

6.	TYPE OF DIFFICULTY	7. SUPPORTING DOCUMENTATION FOR RESEARCH BY USER			
X	APPLICATION SOFTWARE	AVAIL		AVAIL	
	EXECUTIVE SOFTWARE	X	CORE DUMP		OUTPUT FILE-TAPE
	INPUT DATA	X	CONSOLE COPY		DATA FILE-TAPE
	DOCUMENTATION		JOB CONTROLLIST		RPT OUTPUT-TAPE
	HARDWARE	X	SYSLST OUTPUT	X	DOCUMENTATION
	OTHER	X	INPUT FILE-TAPE		OTHER ______
			DISK MODULE		

USACSC FORM 1 FEB 76 53

Replaces edition of 1 May 75 which is obsolete

691

8

IA-01-04-26

I KNOW A BETTER WAY

After you have had an opportunity to observe an AMIS in it's normal operation, you may think of ways to improve it. Additionally, the people who use the output, the functional user, may like to see the output changed or possibly new reports produced. CSC encourages these suggestions and the Army has provided a means of submitting them: the System Change Request (SCR). There are two types of change requests, technical, involving a suggestion for technical ADP reasons and functional, involving a suggestion to change the input or output from the customer's viewpoint. You will be responsible for filling out technical SCR and you should review functional SCR after they have been prepared by the user. You should keep a log of all SCR you have submitted. CSC distributes a monthly SCR status report which you can compare to your SCR log to make sure that your information agrees with CSC's information.

Refer to the sample SCR and we will discuss the blocks you will have to fill out.

Block 1. This will be the address of the functional proponent of the system.

Block 2. The address of your DPA.

Block 3. This number has the same format as the originator number in the incident report.

Block 4. Your name.

Block 5. You can only check routine. The other three blocks are for the functional proponent and CSC.

Block 6. The AMIS name and, if applicable, the program name.

Block 7. If this SCR is the result of an incident, then complete this block. Otherwise leave blank.

Block 8. Self-evident.

Block 9. If the SCR involves system documentation, check the appropriate block.

Block 10. Attach any documents that may help to explain the suggestion and check the appropriate block.

Block 11. Enter the problem as it currently exists in part A. In part B describe the solution you are suggesting in sufficient detail so that it can be clearly understood.

Block 12. Usually you will have to furnish a copy of the SCR to CSC and to your major command MISO.

Block 13. Your name, job and signature.

Block 14. Leave blank.

606

1. TO:
HQDA (DACA-CSS-F)
Pentagon
Washington, DC 20310

2. FROM:
MISO
Fort Harrison, IN 46216

3. ORIGINATOR NO: F01-R111-016

4. POINT OF CONTACT: SP5 Johnson

5. CATEGORY (CHECK ONE):
☐ EMERGENCY ☒ ROUTINE
☐ URGENT ☐ PRIORITY

6. SUBSYSTEM VTAADS
PROGRAM ID P05ANV
VERSION NO

7. INCIDENT ENCOUNTERED
STATION
DATE **TIME**

8. SHORT TITLE: (30 CHARACTERS MAXIMUM INCLUDING SPACES).
Operator MSG in Program P05ANV

9. DOCUMENTATION IDENTIFICATION

A. DPI USER MANUALS	☐	C. EXECUTIVE SOFTWARE	☐
B. FUNCTIONAL USER MANUALS	☐	D. FUNCTIONAL SOFTWARE	☐

10. ATTACHMENTS

A. MAPS	☐	D. FILE PRINTOUTS	☐	G. OUTPUT LISTS	☐
B. CORE DUMPS	☐	E. CONSOLE SHEETS	☒	H. JOB STREAM SEQ.	☐
C. IMPACT STATEMENT	☐	F. DFSR	☐	I. OTHER	☐

11. NARRATIVE:

A. PROBLEM DESCRIPTION: Program P05ANV prints 5 lines of operator messages. This takes excessive console and operator time.

B. RECOMMENDED SOLUTION/ACTION TAKEN:
Delete the unnecessary words and abbreviate the remainder so that only 1 or 2 lines are printed.

12. COPY FURNISHED:
HQ TRADOC MISO
USACSC; CSCS-QAC-M

DATE:

13. PREPARED BY:
SP5 Daniel Johnson
Programmer/Analyst

SGNR: **DATE:**

14. PROPONENT AGENT REVIEW:

A. TYPE OF CHANGE	B. CLASS OF CHANGE	C. EXTENT OF CHANGE
☐ FUNCTIONAL	☐ REGULATORY	☐ MAJOR
☐ TECHNICAL	☐ NON-REGULATORY	☐ MINOR

D. REFERRED TO ARA FOR ANALYSIS (DATE):

E. DISPOSITION:
☐ APPROVED, REQUESTED IMPLEMENTATION:
☐ DISAPPROVED

F. FUNCTIONAL GUIDANCE: ☐ ATTACHED ☐ NOT REQUIRED
☐ TO BE PROVIDED BY

SIGNED: DATE:

DA FORM 4157-R
1 FEB 76

11

694 IA-01-04-26

WHAT NOW?

We hope that you now have an understanding of your relationship in a DPA to the Computer Systems Command. Incident and recommended change reporting will be a fact of Army life to you.

GLOSSARY

1. Army Management Information System (AMIS). This term refers to any software system used in the Army. When used in this text, however, the term refers to standardized software that is designed, tested and programmed at Computer Systems Command and run at many DPA. Sometimes this type of AMIS is called SAMIS (Standard AMIS) or Class A AMIS.

2. Assigned Responsible Agency (ARA). The agency who designs, programs and tests AMIS. CSC is normally the ARA for Class A AMIS.

3. Automatic Digital Network (AUTODIN). A communications link among most Department of Defense activities used for transmitting data from post to post. Data is transmitted electronically (similar to a phone call) and is received either in punched cards or magnetic tape.

4. Computer Systems Command (CSC or USACSC). This is the Army's software design and planning agency. It develops, programs and tests centralized Class A AMIS and plans for the future direction of Army software.

5. Customer Assistance Office (CAO). This is the central contact point that CSC has established for reporting incidents. It is generally manned 24 hours a day. The CAO will take all the information on the incident report and forward it to the responsible programmer for resolution.

6. Data Processing Activity (DPA). Any Army organization which operates data processing equipment. This can range from a unit with just data entry equipment to a unit with several large scale computer systems.

7. Emergency/Urgent Change Package (EUCP). Changes to standard AMIS sent by CSC to resolve operational difficulties. This is the way CSC distributes changes made as a result of incident reports.

8. Functional Proponent. The organization which is responsible for the functions of an AMIS. For example the Deputy Chief of Staff for Personnel is the functional proponent for SIDPERS. Normally HQDA staff offices are the functional proponents for CSC systems.

9. Functional User. The activity which uses the output from an AMIS; usually non-ADP personnel who are customers of the DPA.

10. Incident Report (IR). Reporting a system operational problem, that is, where the system does not work as it should, to CSC via telephone. This is also the written record (USACSC form 53) made in support of the phone call.

11. System Change Package (SCP). A regularly scheduled change to a standard AMIS developed by the ARA and sent to all DPA who run the affected system. An SCP will normally include changes made as the result of SCR's.

12. System Change Request (SCR). A request to either the PA or ARA of a standard system to make a change in the system. If the change is for ADP efficiency, it is called a technical SCR; if it is to change data on initial input or output it is called a functional SCR.

BIN# D-555

HARDWARE/SOFTWARE SYSTEMS EFFICIENCY

SCENARIO

1. SITUATION:

a. As previously stated, you have been assigned to this organization as the Systems Software Analyst, responsible for the operating system and associated software functions.

b. New equipment configuration.

(1) IBM 360 Model 30 - 64K	
(2) 1050 Console Keyboard	1 ea
(3) 2314 Disk w/9 drives	1 ea
(4) 2400 Tape Drive (9 track)	4 ea
(5) 2804 Control Unit	1 ea
(6) 1403 Printer	1 ea
(7) 2540 Card Read/Punch Unit	1 ea
(8) 2821 Control Unit	1 ea
(9) Selector Channel	2 ea
(10) Multiplexer Channel	1 ea

c. Special Features

(1) Floating Point

(2) Decimal

(3) Timer

d. To support

(1) Multiprogramming

(2) COBOL

(3) FORTRAN

(4) PL/I (48 character set)

611

e. Manufacture Supplied Packages

(1) Operating System Routines (Supervisor, Linkage Editor, etc)

(2) Subroutines for FORTRAN and PL/I

(3) Utility Programs

f. User systems now in effect are outlined in the separate systems charts, incl 1 and 2. Memory requirements and run times for each process step are indicated on the chart. Additional programs to be supported are listed in incl 3 and are to be assumed to be stand-alone.

2. REQUIREMENTS

a. Design and generate, on paper, an operating system specifying selected options, core requirements, and total size of your supervisor. Calculate the "Send"Address.

	OPTIONS/SELECTIONS	MAIN STORAGE REQUIREMENTS
SUPV		
SYSTEM=	____________	____________
MPS=	____________	____________
TP=	____________	____________
MICR=	____________	____________
ASCII=	____________	____________
AP=	____________	____________
EU=	____________	____________
ERRLOG=	____________	____________
MCRR=	____________	____________
CONFG		
MODEL=	____________	____________
SP=	____________	____________
DEC=	____________	____________
FP=	____________	____________
TIMER=	____________	____________
PORT=	____________	____________
STDJC		
DECK=	____________	____________
LIST=	____________	____________
LISTX=	____________	____________

613

	Options/Selections	Main Storage Requirements
SYM=		
XREF=		
ERRS=		
CHARSET=		
LOG=		
DUMP=		
LINES=		
DATE=		
SPARM=		
FOPT		
OC=		
IT=		
PC=		
TEB=		
TEBV=		
EVA=		
SKSEP=		
CE=		
JA=		
JALIOCS=		
PTO=		
IDRA=		
OLTEP=		
RETAIN=		
PCIL=		
CBF=		

614

	Options/Selections	Main Storage Requirements
CCHAIN=	________	________
TRKHLD=	________	________
AB=	________	________
WAITM=	________	________
DASDFP=	________	________
SYSFIL=	________	________
PIOCS	________	________
SELCH=	________	________
BMPX=	________	________
CHANSW=	________	________
TAPE=	________	________
MRSLCH=	________	________
IOTAB	________	________
BGPGR=	________	________
F1PGR=	________	________
F2PGR=	________	________
JIB=	________	________
CHANQ=	________	________
IODEV=	________	________
		________
TOTAL		________

Send Address: SEND ________________

615

b. List available channels and distribution of I/O devices:

Type Channel	Channel Number Assigned	Devices

c. Specify size of required partitions:

ALLOC F1= ______K,F2=_______K

616

d. Prepare a schedule of jobs for a Tuesday in which the payroll checks are to be run. Shift starts at 0700 hrs and we run two shifts of eight hours each. Indicate Program Number, Time to be Run and Partition. Schedule should be such that the first job on the schedule is the first job to be run, etc.

Program Number	Time to be Run	Partition

617

e. List contents of each library in specific terms, by routine name or program number.

(1) Core Image Library

(2) Relocatable Library

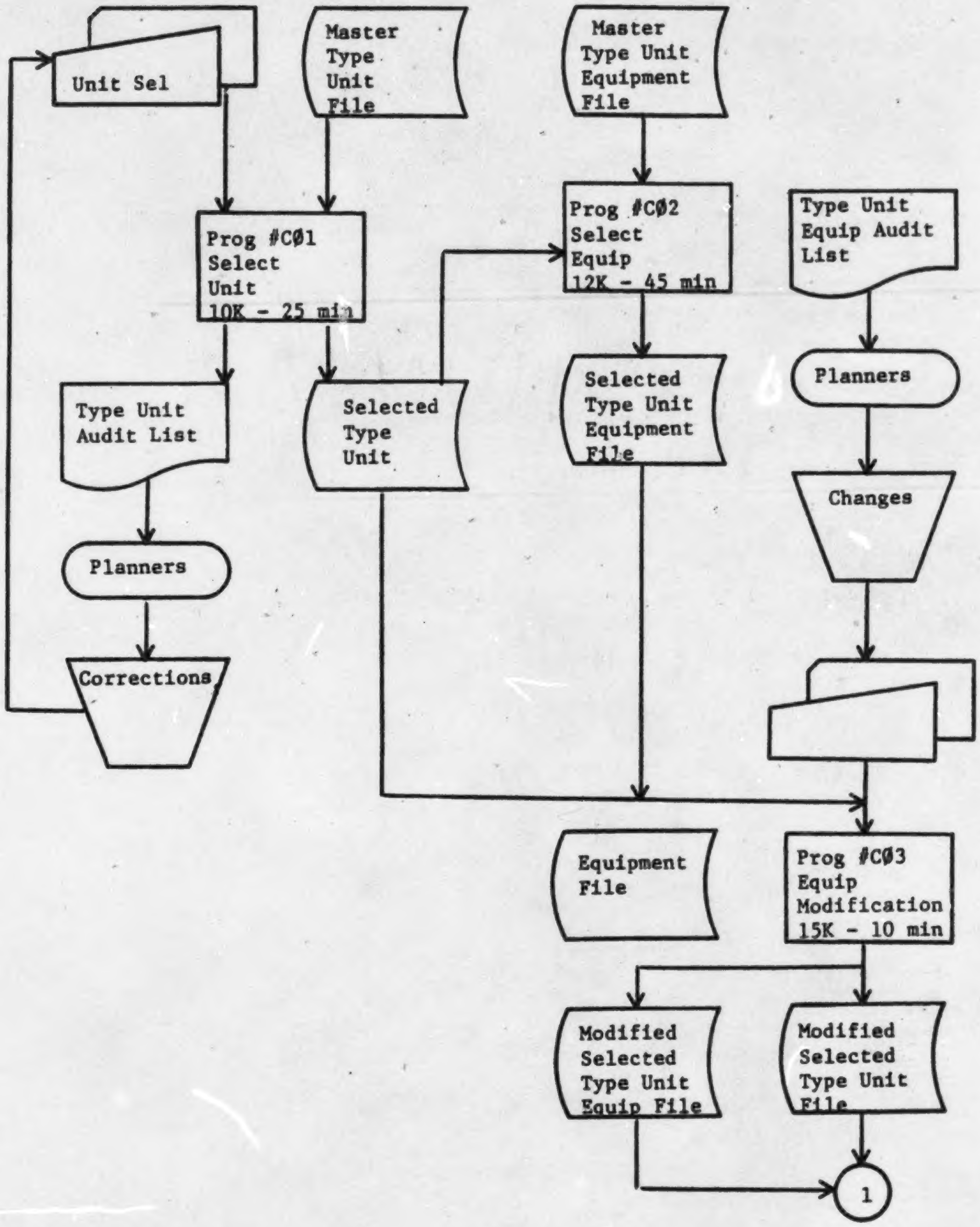
(3) Source Statement Library

f. Does your system require "Spooling"? Yes ______ No ________ Why?

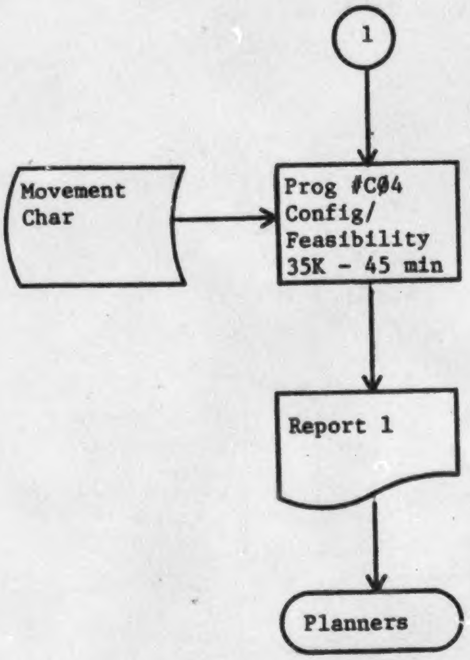
619

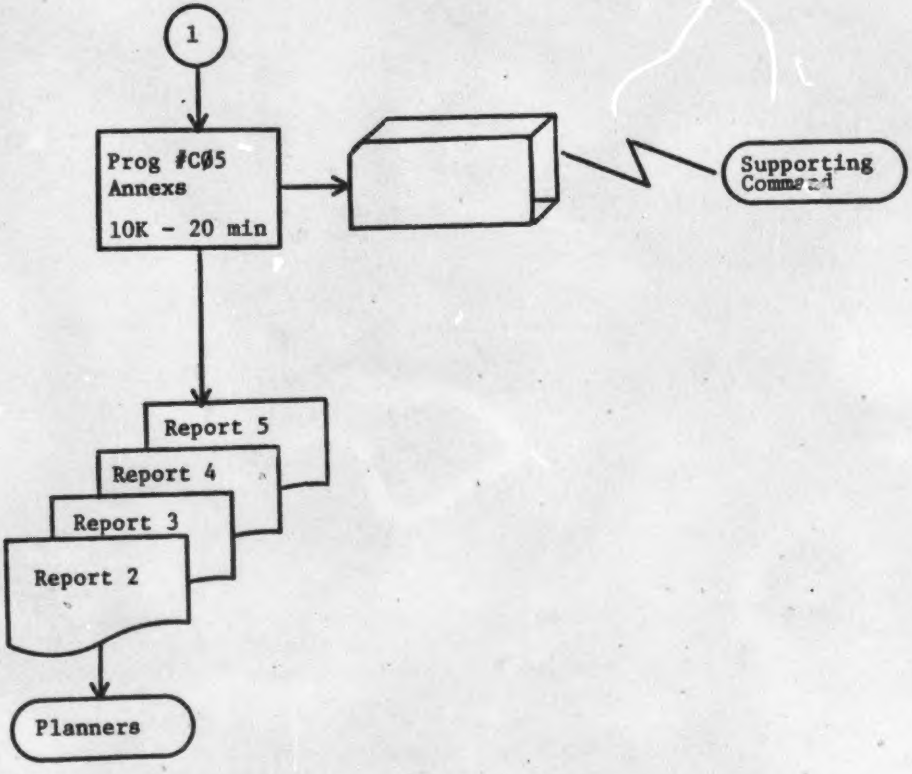
CONTINGENCY PLANNING

FREQ: As Required



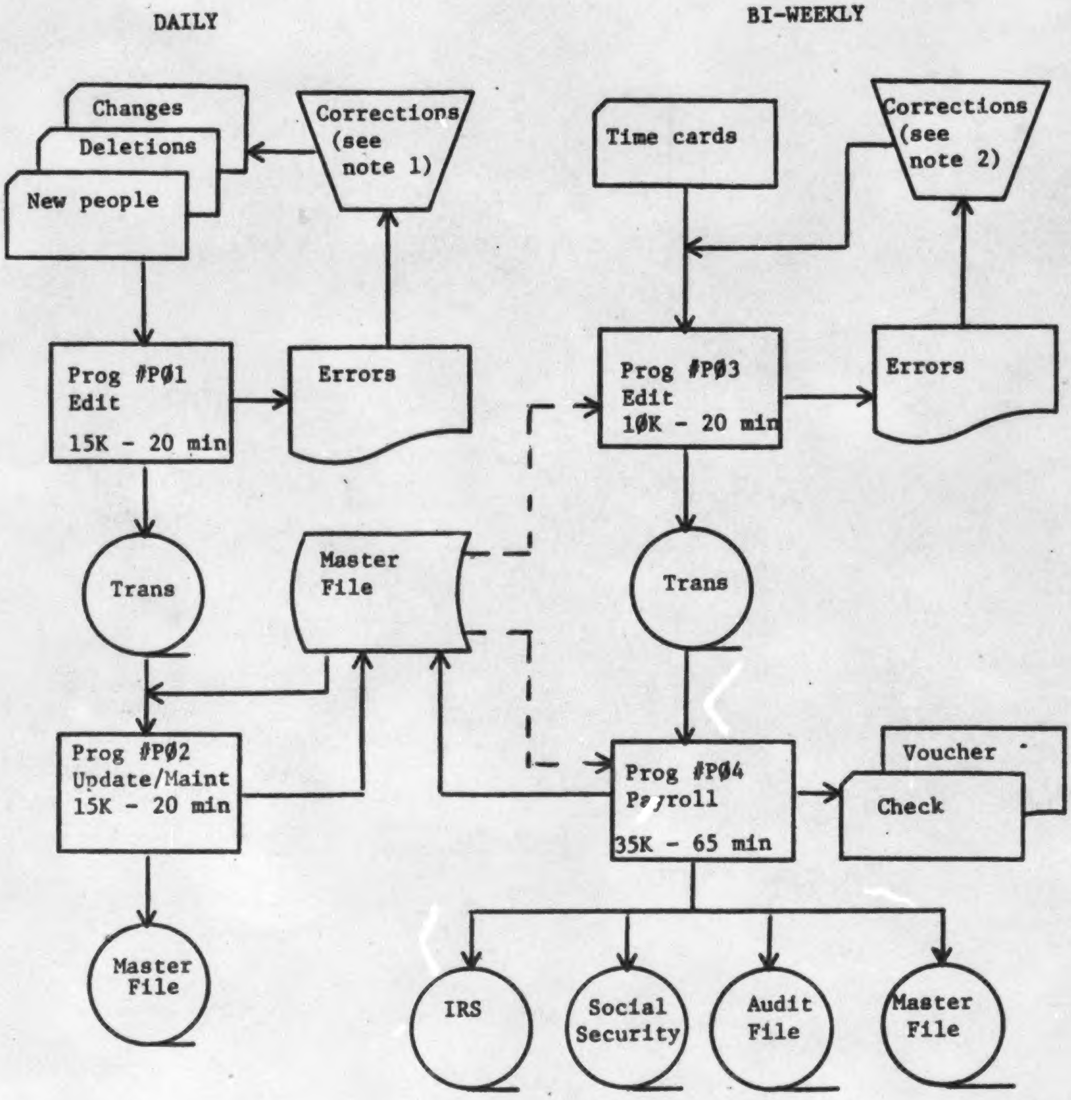
Unit Sel
Master Type Unit File
Master Type Unit Equipment File
Prog #C01 Select Unit 10K - 25 min
Prog #C02 Select Equip 12K - 45 min
Type Unit Equip Audit List
Type Unit Audit List
Selected Type Unit
Selected Type Unit Equipment File
Planners
Planners
Changes
Corrections
Equipment File
Prog #C03 Equip Modification 15K - 10 min
Modified Selected Type Unit Equip File
Modified Selected Type Unit File
1


1
Movement Char
Prog #C04 Config/ Feasibility 35K - 45 min
Report 1
Planners


1
Prog #C05 Annexs 10K - 20 min
Supporting Command
Report 5
Report 4
Report 3
Report 2
Planners

Incl 1 - pg 2

CIVILIAN PAYROLL


DAILY
BI-WEEKLY
Changes
Deletions
New people
Corrections (see note 1)
Prog #P01 Edit 15K - 20 min
Errors
Trans
Master File
Prog #P02 Update/Maint 15K - 20 min
Master File
Time cards
Corrections (see note 2)
Prog #P03 Edit 10K - 20 min
Errors
Trans
Prog #P04 Payroll 35K - 65 min
Voucher
Check
IRS
Social Security
Audit File
Master File

Note 1: The errors are to be corrected and the corrections inputed into the next day's cycle, except those errors of the last cycle for the period are to be corrected immediately and the edit rerun. All errors must be corrected before the Payroll can be run.

Note 2: The errors must be corrected immediately and the edit rerun until no errors exist. Only then will the payroll checks be run.

Incl 2

Incl 3

ADDITIONAL PROGRAMS

PROG NR	SIZE K	FREQ	TIME REQD	RUN TIME	INPUT FILES			OUTPUT FILES			
					TAPE	DISK	READER	TAPE	DISK	PUNCH	PRINTER
A01	27	Daily	1000 hrs	10	2	1	X		1		1
A02	10	Daily	0800 hrs	3	1	1	X		1		1
A03	33	Daily	2400 hrs	25	2	3	X	2	2		2
A04	20	Daily	2400 hrs	57	1	2				2	1
A05	15	Daily	1200 hrs	15		3	X				3
A06	25	Daily	2400 hrs	30	2		X	1		1	2
A07	35	Daily	2400 hrs	240	1	4	X		4	2	3
A08	34	Tuesday	2400 hrs	65	2	4			2		3
A09	29	Friday	2400 hrs	30		2	X	1	1		1
A10	15	Monday	2400 hrs	27	1	3			2	1	2
A11	21	Wed	2400 hrs	32	2	4	X	2			4
A12	17	Thur	2400 hrs	19		2		1		1	
A97		Daily		120	Compile and Test Time						
A98		Daily		60	Software Maintenance/Software Analyst						
A99		Daily		60	Systems Maintenance/Manufactures Rep						
Spooling	18	If determined to be required									

13

USAF PP 190-30 7/74

622